9-1-2018

Simplex Control Methods for Robust Convergence of Small Unmanned Aircraft Flight Trajectories in the Constrained Urban Environment

Michael D. Zollars

Simplex Control Methods for Robust Convergence of Small Unmanned Aircraft Flight Trajectories in the Constrained Urban Environment

DISSERTATION

Michael D. Zollars, Lt. Colonel, USAF

AFIT-ENY-DS-18-S-078

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENY-DS-18-S-078

SIMPLEX CONTROL METHODS FOR ROBUST CONVERGENCE OF SMALL UNMANNED AIRCRAFT FLIGHT TRAJECTORIES IN THE CONSTRAINED URBAN ENVIRONMENT

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Michael D. Zollars, B.S., M.S.

Lt. Colonel, USAF

12 July 2018

AFIT-ENY-DS-18-S-078

SIMPLEX CONTROL METHODS FOR ROBUST CONVERGENCE OF SMALL UNMANNED AIRCRAFT FLIGHT TRAJECTORIES IN THE CONSTRAINED URBAN ENVIRONMENT

DISSERTATION

Michael D. Zollars, B.S., M.S.
Lt. Colonel, USAF

Committee Membership:

Dr. R. G. Cobb
Chairman

Dr. D. R. Jacques
Member

Dr. W. P. Baker
Member

Abstract

Constrained optimal control problems for Small Unmanned Aircraft Systems (SUAS) have long suffered from excessive computation times caused by a combination of constraint modeling techniques, the quality of the initial path solution provided to the optimal control solver, and improperly defining the bounds on system state variables, ultimately preventing implementation into real-time, on-board systems. In this research, a new hybrid approach is examined for real-time path planning of SUAS. During autonomous flight, a SUAS is tasked to traverse from one target region to a second target region while avoiding hard constraints consisting of building structures of an urban environment. Feasible path solutions are determined through highly constrained spaces, investigating narrow corridors, visiting multiple waypoints, and minimizing incursions to keep-out regions. These issues are addressed herein with a new approach by triangulating the search space in two-dimensions, or using a tetrahedron discretization in three-dimensions to define a polygonal search corridor free of constraints while alleviating the dependency of problem specific parameters by translating the problem to barycentric coordinates. Within this connected simplex construct, trajectories are solved using direct orthogonal collocation methods while leveraging navigation mesh techniques developed for fast geometric path planning solutions. To illustrate two-dimensional flight trajectories, sample results are applied to flight through downtown Chicago at an altitude of 600 feet above ground level. The three-dimensional problem is examined for feasibility by applying the methodology to a small scale problem. Computation and objective times are reported to illustrate the design implications for real-time optimal control systems, with results showing 86% reduction in computation time over traditional methods.

Acknowledgements

The last three years have been unlike any other in my short life. The challenges, stress, and unknowns of the doctoral program were frustrating at times and left me wondering if I would ever get through. Over the time, there were so many people that helped me along the way and deserve the most gracious thanks. First and foremost, my advisor, Dr. Richard Cobb. He has shown confidence in me and provided the right mix of direction and laughter. Weekly meetings consisted of a range of topics from sports, to family, to everything I needed to maintain the pace on my research. I could not have picked a better mentor! Likewise, I want to thank my research committee members, Dr. William Baker and Dr. David Jacques for their continued support and guidance. Next, I would not have a research topic if it were not for Dr. David Grymin. Always willing to meet and give advice and direction helped me fill the holes in my work and provided a solid foundation for my research. The students of the class of 18S made the penthouse an enjoyable place where distractions were needed at times, and the prior graduates of the Aeronautical department in control and optimization theory provided so much guidance along the way, namely Dr. Ryan Carr and Dr. Clay Humphreys. Thanks to my parents for giving me the desire to pursue the degree and the work ethic to get through it. Finally, and most important, my family has joined me for this three year adventure. My wife has picked up the slack during long nights of work and my kids have been entertained as they see the complexities of aircraft path planning. Thank you for your patience and understanding. Thank you to all that have helped me progress through this doctoral program.

Michael D. Zollars

Table of Contents

Page

Abstract iv

Acknowledgements v

List of Figures ix

List of Tables xii

List of Abbreviations xiii

I. Introduction 1

1.1 Motivation 3
1.2 Research Questions, Scope, and Tasks 5
Research Questions 5
Research Scope 6
Research Tasks 8
1.3 Assumptions and Limitations 9
Dynamics and UAS Model 9
CONOPS 10
1.4 Research Methodology 10
1.5 Expected Contributions 13
1.6 Document Outline 13

II. Literature Review 15

2.1 Introduction 15
2.2 Optimal Control 15
The Optimal Control Problem 17
2.3 Constrained Trajectory Optimization 22
Heuristic Methods 28
2.4 Fast Geometric Path Planning 29
2.5 Discretization Methods 30
Delaunay Triangulation 31
Constrained Delaunay Triangulation 32
Path Search Algorithms 34
Triplanner Toolkit 38
Barycentric Coordinate Frame 40
2.6 Summary 42

Page

III. Methodology . . . 43

3.1 Overview . . . 43
3.2 Software . . . 44
3.3 Triplanner Toolkit . . . 45
3.4 Tetrahedron Discretization . . . 46
3.5 Optimal Control Problem . . . 49
GPOPS-II Input Parameters . . . 50
Coordinate Transformation . . . 51
GPOPS-II Phased Solution . . . 53
SUAS Dynamics . . . 54
Objective Function . . . 55
Dynamic Constraints . . . 56
Path Constraints . . . 59
Integral Constraints . . . 60
Event Constraints . . . 61
Bounds . . . 62
Algorithm Development . . . 63
3.6 Summary . . . 67

IV. Constrained Optimization Approach . . . 68

4.1 Overview . . . 68
4.2 Constraint Analysis Simulation Overview . . . 68
4.3 Constraint Analysis Simulation Results . . . 73
4.4 Two-Dimensional Optimal Control . . . 75
Optimal Control Problem . . . 75
4.5 Two-Dimensional Scenario . . . 78
4.6 Two-Dimensional Conculsions . . . 83
4.7 Three-Dimensional Optimal Control Problem . . . 85
Optimal Control Problem . . . 85
4.8 Three-Dimensional Scenario . . . 88
Single Phase Formulation . . . 89
Simplex Formulation . . . 91
Three-Dimensional Results . . . 92
4.9 Three-Dimensional Conclusions . . . 96

V. Variations to the Two-Dimensional Problem . . . 98

5.1 Overview . . . 98
5.2 Optimal Control Problem . . . 98
5.3 Waypoint Following . . . 101
Waypoint Scenario Development . . . 101
Waypoint Scenario Results . . . 104
Waypoint Scenario Conclusions . . . 107

Page

5.4 Aircraft Keep-Out Regions . . . 109
Keep-Out Region Scenario Development . . . 109
Keep-Out Region Scenario Results . . . 111
Keep-Out Region Conclusions . . . 114
5.5 Wind Analysis . . . 115
Wind Analysis Algorithm Development . . . 115
Wind Analysis Scenarios Results . . . 116
Wind Analysis Conclusions . . . 119
5.6 Contingency Planning . . . 120
Contingency Operation Algorithm Development . . . 121
Contingency Operation Results . . . 122
Contingency Operation Conclusions . . . 124
5.7 Summary . . . 125

VI. Conclusions and Recommendations . . . 126

6.1 Conclusions . . . 126
6.2 Contributions . . . 129
6.3 Potential Future Research . . . 130
6.4 Summary . . . 134

Appendix A. Presentations . . . 136

Appendix B. Dynamic Systems Controls Conference 2017 . . . 138

Appendix C. SciTech Information Systems Conference 2018 . . . 149

Appendix D. Journal of Aeronautics & Aerospace Engineering . . . 166

Appendix E. Aerospace Conference 2018 . . . 175

Appendix F. American Controls Conference 2018 . . . 184

Appendix G. Unmanned Systems Journal 2018 . . . 191

Bibliography . . . 202

List of Figures

Figure		Page
1	Non-Linear Feedback Control	7
2	Data Flow Chart	12
3	SUAS Capabilities	12
4	SUAS Optimal Control Topology & Chapter Layout	14
5	Closed Curve Superellipse	24
6	Two-dimensional Superellipse	25
7	Stiffness Transition of Sigmoid Function	26
8	Dirichlet Tessellation and Delaunay Triangulation	31
9	Comparison of Constrained and Non-Constrained Delaunay Triangulation	33
10	RRT Search Algorithm	37
11	Funnel Algorithm Illustration	40
12	Triplanner Algorithm Illustration	46
13	Three Dimensional Constraint Map	47
14	Three Dimensional Discretization	48
15	Barycentric Coordinate Frame for a Tetrahedron	51
16	Event Constraints through Simplexes	61
17	Algorithm Flow Chart	63
18	Triplanner Solution	64
19	Triplanner Interpolated Path Solution	65
20	Initial State and Control Vectors	66
21	Simple Shape Skinny Constraint Functions	70
22	Superquadric Constraint Functions	70

Figure Page

23 CDT Path Solution & Control; Mid-point Guess 72

24 Constraint Analysis Triplanner Tool-kit Solution 72

25 Hybrid Path Solution & Control 73

26 Discretized Mesh of Downtown Chicago 79

27 Two-Dimensional Triplanner Toolkit Path Solution 80

28 Two-Dimensional Optimal Path Solution; Mid-Point Guess 81

29 Two-Dimensional Optimal Path Solution & Control; Mid-Point Guess 82

30 Two-Dimensional Hybrid Path Solution 82

31 Two-Dimensional Hybrid Path Solution & Control 83

32 Three-Dimensional Constraint Map 89

33 Three-Dimensional Superellipsoid Constraint Map 91

34 Three-Dimensional Simplex Search Corridor & Initial Guess 92

35 Three-Dimensional Optimal Path Solution Comparison 93

36 Three-Dimensional Optimal State & Control 94

37 Three-Dimensional Path & Constraint Comparison 95

38 Multiple Waypoint Constraint Map 102

39 Multiple Waypoint Interval Development 103

40 Multiple Waypoint Triplanner Solution 104

41 Multiple Waypoint Hybrid Solution 106

42 Multiple Waypoint Hybrid Solution State & Control 107

43 Keep-Out Region Constraint Map 109

44 Keep-Out Region Triplanner Solution 111

Figure Page

45 Keep-Out Region Hybrid Solution 112

46 Keep-Out Region Hybrid Solution State & Control 113

47 Min Time Flight with Varying Wind Magnitude 117

48 Varying Wind Magnitude, State Trajectories 118

49 Min Time of Flight with Varying Wind Direction 118

50 Varying Wind Direction, State Trajectories 119

51 Contingency Algorithm Development 120

52 Contingency Algorithm Analytical Results 123

53 Contingency Algorithm Optimal Path Solution 123

54 Contingency Algorithm Optimal Control Demonstration 124

55 Triplanner Multiple Path Solutions 131

List of Tables

Table		Page
1	Optimal Control Aircraft Models	55
2	Constraint Analysis Simulation Results	74
3	GPOPS-II User Settings, 2D Scenarios	80
4	Triplanner Results	84
5	Two-Dimensional Simulation Results	84
6	GPOPS-II User Settings, 3D Scenarios	88
7	Three-Dimensional Simulation Results	94
8	Triplanner Interval Solutions	105

List of Abbreviations

Abbreviation		Page
USAF	United States Air Force	1
UAS	Unmanned Aircraft Systems	1
CONOPS	Concepts of Operations	1
CAS	Close Air Support	2
SCAR	Strike Coordination and Reconnaissance	2
SUAS	Small Unmanned Aircraft System	2
ISR	Intelligence, Surveillance, and Reconnaissance	2
MUM-T	Manned Unmanned Teaming	2
TOBS	Tactical Off-Board Sensing	3
AFSOC	Air Force Special Operations Command	3
AFRL	Air Force Research Laboratory	3
NLP	Non-Linear Program	4
CSC	Connected Simplex Corridor	5
DP	Dynamic Programming	7
AFIT	Air Force Institute of Technology	15
HBVP	Hamiltonian Boundary-Value Problem	19
IPOPT	Interior Point Optimizer	23
SNOPT	Sparse Nonlinear Optimizer	23
CDT	Constrained Delaunay Triangulation	32
RRT	Rapidly exploring Random Tree	37
LCT	Local Clearance Triangulation	38
AGL	Above Ground Level	78

SIMPLEX CONTROL METHODS FOR ROBUST CONVERGENCE OF SMALL UNMANNED AIRCRAFT FLIGHT TRAJECTORIES IN THE CONSTRAINED URBAN ENVIRONMENT

I. Introduction

In the last 25 years, the United States Air Force (USAF) has seen the roles and responsibilities of Unmanned Aircraft Systems (UAS) continue to grow and be recognized as critical assets across all levels of joint and multinational command [1]. Over this time, UAS missions have been categorized into the dull, the dirty, and the dangerous missions as laid out in the Unmanned Systems Integrated Roadmap for fiscal year 2013 - 2038 [2]. The dull missions are defined as those mundane tasks that may need to be repeated continuously, the dirty missions are those that interact with hazardous materials, and the dangerous missions involve high risk activities that threaten the integrity of the aircraft and its crew. Developing UAS that can meet these mission needs creates a challenge for research and development to field unmanned systems that are affordable, flexible, interoperable, integrated, technologically advanced, and capable of providing decisive force on the battlefield.

The Joint Concepts of Operations (CONOPS) for Unmanned Aircraft Systems describes the most advantageous quality of the UAS as the ability to significantly reduce the risk to human life for repetitive or dangerous missions [1]. Further, due to the early successes of autonomous UAS in military operations, a 2003 congressional report [3] labeled the UAS as a transformational technology that could change the way wars are fought and won. Now, UAS are recognized as critical assets across all levels of joint and multinational command, and the demand for the capabilities they

can provide today and into the future will continue to grow [1].

Over the past two decades, the documentation that governs the roles and responsibilities of the UAS has continued to be refined. The CONOPS document defines the tasks for which the UAS shall support military activities. These include target acquisition/marking, delivery of onboard precision-guided ordnance, tactical assessment, and battle damage assessment. The CONOPS further defines dynamic targeting situations where a UAS would augment Close Air Support (CAS), Strike Coordination and Reconnaissance (SCAR), reconnaissance, air interdiction, and personal recovery [1]. Each of these tasks may require different levels of autonomy. Further, the Deputy Chief of Staff for Intelligence has advocated for automation in the Small Unmanned Aircraft System (SUAS) Flight Plan 2016-2036 [4] stating that onboard automation is meant to streamline systems, sensors, and analytical tasks essential for the exploitation of actual intelligence to augment and enhance human capabilities.

Along with the documentation, history has shown the advancement in the roles and responsibilities of the UAS. At the start of the modern era in 1995, the USAF first flew the RQ-1 Predator, which was predominantly used for reconnaissance missions. By 2002, the Desert Hawk become the first SUAS. The mission, defined by the needs of the war-fighter, shifted from just reconnaissance to Intelligence, Surveillance, and Reconnaissance (ISR) along with target recognition [4]. As additional aircraft joined the fleet, the success of the SUAS in military operations illustrated the surprisingly fast growth of the roles and responsibilities performed by unmanned aircraft. In a 2003 report to Congress, the roles of the SUAS continued to increase as the report initiated the call for unmanned aircraft to team up with manned aircraft to carry out military operations [3]. Manned Unmanned Teaming (MUM-T) has since become a top objective in SUAS development moving into the future.

1.1 Motivation

In the SUAS Flight Plan 2016-2036, the CONOPS describes the operations of small, agile unmanned aircraft teaming up with manned aircraft to accomplish tactical to strategic level mission objectives [4]. These missions are taken a step further in the CONOPS from MUM-T to loyal wingman operations where the efficiencies of the manned flight are augmented with a subordinate SUAS to increase the overall capability of the flight operations. The document describes the Tactical Off-Board Sensing (TOBS) concept for loyal wingman where a SUAS is teamed with a manned aircraft for ISR or strike missions. Coordination between the two aircraft remain for the duration of the mission, specifically as the SUAS explores the environment for target data updates, thus reducing the burden and risk of life to the human operators of the manned aircraft.

The Air Force Special Operations Command (AFSOC) is working with the Air Force Research Laboratory (AFRL) to improve AFSOC's targeting capabilities in poor weather and other challenging conditions through TOBS. AFRL's vision for autonomy, science, and technology for the year 2020 is to obtain "intelligent machines seamlessly integrated with humans - maximizing mission performance in complex and contested environments" [5]. The Autonomy Science and Technology Strategy signed in 2013 states four goals aimed to achieve this vision [5]. The third goal, "Ensure operations in complex, contested environments" refers directly to the TOBS effort.

The TOBS problem is executed in two phases. In order to give a host aircraft a better view of the battlefield, a SUAS is deployed to locate target coordinates at a low altitude, possibly below weather or other obstructions. In the first phase, the SUAS acts like a sensor for the host aircraft, remaining in the target region until the operator engages on the target or dismisses the effort. The second phase consists of the SUAS traveling from the initial target region to a second target region. The

required flight path in this phase must be determined autonomously, with a real-time solution while avoiding terrain, man-made constraints, and airspace keep-out regions. AFSOC has stated their desire of the system as "a software driven autopilot...you tell it where you want it to go and where you want it to orbit and then it goes there" [6].

The solution to the first phase of the TOBS problem was evaluated in Heidlauf's work [7] and continues to be developed at AFRL's Power and Control's Division. AFRL's interpretation of the second phase of the TOBS problem is to guarantee a solution for the flight path beginning at the terminal state of phase one and concluding at the second target location. Flight paths are to be determined in near real-time using optimal control techniques.

There are two common challenges that prove to be problematic when using optimal control software. First, the convergence to a solution is not always guaranteed. Second, the time required to achieve a solution can vary greatly. Both of these issues can be attributed to the problem formulation, the implementation of the constraints, and the initial guess that is used to seed the Non-Linear Program (NLP) solver. To overcome these issues, insight will be taken from developments in fast geometric path planning algorithms where simplex discretization techniques are used to quickly find feasible paths through constrained regions.

Initially, the two-dimensional environment will be explored for feasibility of the solution method. The mission scenario will be restricted to a constant altitude and path constraints will be specified. Fast geometric path planning techniques will be used to discretize the domain and generate an initial path for the vehicle that will be used to seed the optimal control solver in an attempt to improve the computational speed and accuracy of the NLP. Expanding the scenario to three-dimensional space is challenging as there is not a simplex discretization method that can immediately be leveraged as in the two-dimensional case. Therefore, an analysis will be conducted

to determine feasible methods to expand the problem domain using a triangulation or tetrahedron technique, referred to as a Connected Simplex Corridor (CSC) in this document, in order to acquire optimal path solutions through an urban environment.

1.2 Research Questions, Scope, and Tasks

The TOBS program is the program of record for this research and is defined with a host aircraft and a subordinate aircraft. Throughout this work, the SUAS will be defined as the subordinate aircraft either air or ground deployed and in constant communication with the host aircraft. Throughout the duration of phase two, path planning and control shall be completed onboard the SUAS with no guidance from the host aircraft other than to inform the SUAS on target location and constraint regions.

Research Questions.

Hypothesis: The existence of a feasible flight path through a constrained environment can be determined quickly and efficiently (in near real-time) with a hybrid method combining optimal control direct orthogonal collocation methods with fast geometric path planning techniques.

The research questions related to this hypothesis are:

1. How do you formulate the two-dimensional optimal control problem for optimal trajectories in complex urban environments?

2. Can the computational speed and robustness of convergence to the two-dimensional optimal control problem be improved by formulating the problem within a CSC construct using fast geometric path planning techniques?

3. How do dynamic constraints and constant wind fields affect the flight path

solution acquired from a simplex discretization?

4. How can the three-dimensional problem be formulated using simplex or polyhedronal discretization techniques?

Research Scope.

The work outlined in this research will commence at the terminal state of phase one of the TOBS problem and will continue through the duration of phase two. Phase two will terminate at a specified location for the second target or on a defined orbit around the target.

The primary scope of this research will be limited to constrained trajectory optimization motivated by increasing computation speeds, accuracy, and convergence rates of the optimal control software. Initially, the problem will be limited to two-dimensional flight with constraints representing an urban setting. To provide a somewhat realistic scenario, constraints will be determined from image and building information of a portion of downtown Chicago, USA. Methods and limitations to extend the two-dimensional techniques to three dimensions will be explored. The TOBS aircraft will be ground or air launched and assumed to have constant communication with the host aircraft, however control and path generation must be accomplished onboard the SUAS. The aircraft hardware and sensor packages will be consistent with the phase one design and therefore is beyond the scope of this research.

Figure 1 describes a typical feedback control system for non-linear control. Given a commanded input of aircraft Euler angles, position, and desired terminal position, a smooth and achievable aircraft trajectory is generated in the open loop. The closed loop control consists of controllers, vehicle dynamics, exterior disturbances such as wind gusts, and sensor data. The feedforward controller generates a set of controls based on the ideal set of dynamics. The errors that are associated with mis-modeled

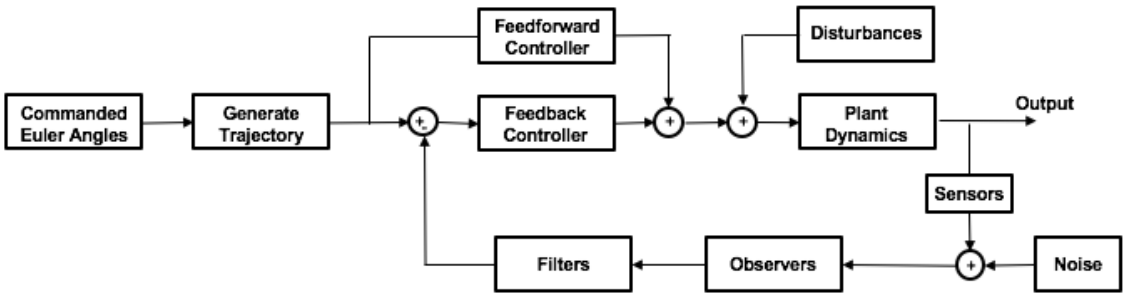

Figure 1. Non-Linear Feedback Control

dynamics, exogenous inputs, and sensor noise are accounted for within the feedback controller. In order for the feedback control to have full controllability, observers and filters are incorporated into the feedback loop such that estimates of the full state can be achieved [8]. The work herein presents methodologies for generating aircraft trajectories, shown in the "Generate Trajectory" block in Figure 1, onboard the SUAS in near real-time based on a priori knowledge of flight parameters and disturbances. The strength and direction of wind fields and the motion of dynamic constraints are assumed known such that path trajectories may be acquired in realistic environments. Modeling the closed loop to maintain the control parameters along the desired path while accounting for mis-modeled dynamics, exogenous inputs, and sensor noise is outside the scope of this research.

The optimal control used to find the feasible path can be formulated and solved using many different methods. This research will explore direct orthogonal collocation methods which have proven to be effective in computing real-time solutions to optimal control problems [9]. Other methods such as Dynamic Programming (DP) and heuristic search algorithms will be used to find feasible regions of the optimal solution. Fast geometric path planning methods will be explored as a method for discretization, allowing for a phased solution through a CSC.

Real-time solutions will be evaluated based on a comparison between different

models of the constrained environment. The actual time required to solve the flight trajectory will be dependent on several factors such as the algorithm chosen, computer processor speed, and memory allocation.

Research Tasks.

With the research objectives defined above, the following seven tasks will be accomplished.

1. Evaluate path planning constraint equations and the impact different formulations have on the optimal control problem.

2. Identify a mission scenario and formulate the minimum time constrained optimal control problem with polygonal static constraints in an urban environment.

3. Implement CSC techniques for optimal path planning to multiple waypoints within a highly constrained urban environment.

4. Minimize incursions to keep-out regions incorporated into a constrained urban environment while implementing CSC techniques.

5. Determine the effects of constant wind fields on the feasible flight paths acquired with CSC methods in optimal control.

6. Explore contingency operations for dynamic path obstructions with CSC techniques.

7. Examine the CSC methodology to extend the TOBS two-dimensional simplex control problem to the third-dimension.

1.3 Assumptions and Limitations

The assumptions and limitations made within the scope of this research are required to sufficiently bound the tasks such that they remain tractable within the employed numerical techniques. The tools employed to acquire computationally efficient algorithms for SUAS path trajectories provide a simplex search space found with a greedy A* search algorithm. Therefore, this research focuses on determining feasible solutions that satisfy the dynamic constraints consistently and efficiently as opposed to the necessity of arriving at the global optimal trajectory for each simulation.

Dynamics and UAS Model.

The SUAS is modeled as a point mass and initially all external disturbance forces on the vehicle will be excluded. As techniques are discovered that guarantee convergence of the flight path trajectory, constant wind fields will be implemented to analyze the affects on the feasible solution. This work will be accomplished in simulation only within MATLAB® on a personal computer.

For the two-dimensional tasks, the SUAS will be modeled with five states consisting of the vehicle's x-position, y-position, heading angle, heading angle rate, and velocity. The control for the SUAS will consist of the change in heading angle rate and acceleration. The three-dimensional problem will consist of a five state model consisting of the vehicle's x-position, y-position, z-position, heading angle, and pitch angle. The control for the three-dimensional scenarios will consist of the heading angle rate, pitch rate, and velocity. The SUAS performance specifications for vehicle parameters was determined in coordination with AFRL such that a tractable scenario could be developed. Currently there is not a specific vehicle designed for this research, however, design characteristics will fall within a range from Group 1 to Group 2 unmanned aircraft [1, 10].

CONOPS.

For the TOBS mission, SUAS can be hand launched from a ground unit or air launched from a tube located on the host aircraft. Depending on which technique is employed it could drastically affect the size, weight, and performance characteristics of the vehicle. This research effort will assume the vehicle is ground launched and therefore could have an extensive flight time allowing the vehicle to travel large distances between target region one and target region two. The constraints encountered will model an urban canyon applicable within the TOBS mission scenario with realistic vehicle parameters and mission times, defined in Chapter IV.

1.4 Research Methodology

Two primary tools are used in this research. The first is a MATLAB® based software package for solving general purpose optimal control problems, GPOPS-II. This is a general purpose optimal control software package that is used to solve nonlinear optimal control problems using direct orthogonal collocation and Gaussian quadrature methods [11]. Required input parameters to GPOPS-II will be standardized for any TOBS mission and will be solved through a phased approach that is built from a CSC technique.

The second tool is the Triplanner toolkit used in the two-dimensional scenarios. With the Triplanner toolkit, a discretized simplex mesh is generated based on the constraints in the environment, rather than the size of the space. For each simplex, the vertex points provide coordinates for which any point within the simplex can be determined using a barycentric coordinate system. Through the discretization technique, the constraints become polygonal regions that can be eliminated from the search space. Using fast geometric path planning algorithms, a feasible path can be found consisting of a series of traversed simplexes, formed from either a triangular

or tetrahedron mesh, creating a "search corridor", or CSC. This CSC along with the feasible path solution can be used to seed a NLP within an optimal control solver such as GPOPS-II to determine the optimal path through the defined space. When connecting multiple results from Triplanner, an interval is defined as one CSC connecting two defined points. When accomplishing a path across multiple waypoints, several intervals will be combined to determine the complete optimal solution.

Leveraging work performed in fast geometric path planning allows for feasible path solutions that are in close proximity to the optimal solution. The geometric path solutions alone are ideal for computer games and video software where a level of realism is desired, however they are not intended for designing rate limited control variables for air vehicle path planning and therefore including real vehicle dynamics and control through optimal control software is a key component to the methodology.

Figure 2 describes the basic components of the algorithm required to implement this methodology. The algorithm is initiated by defining mission specific parameters. These include the initial and terminal points of the mission, the vertex points of the domain space, the vertex points of each building constraint that must be avoided, and the vehicle specific bounds on the SUAS state and control variables. These inputs are given to a fast geometric path planner where the search space is discretized, a CSC is selected, and an initial path solution is determined. Next, the data is conditioned to formulate a complete guess vector for the time, states, and control. These vectors are partitioned into each simplex of the CSC. The connectivity matrix is constructed to provide the proper order of the channel, analytical calculations for the methodology, and define state values across simplex bounds. This data is provided to the optimal control solver GPOPS-II where a transformation is performed to the barycentric coordinate system so that a phased approach can implemented in solving for the optimal solution.

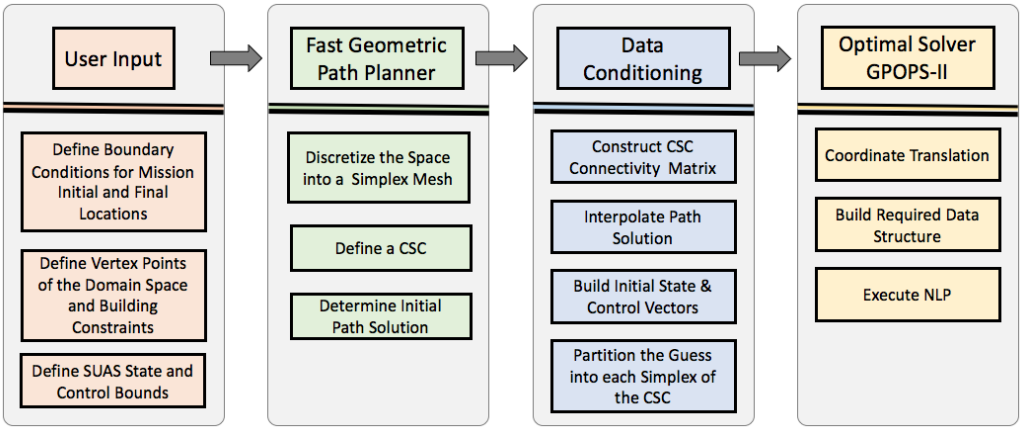

Figure 2. Data Flow Chart

Through this fast discretization process, constraints will be eliminated, an initial guess used to seed the NLP will be determined quickly, and the problem will be formulated in GPOPS-II such that a feasible solution to the control problem can be calculated for onboard solutions. Figure 3 illustrates the capabilities demonstrated in this work.

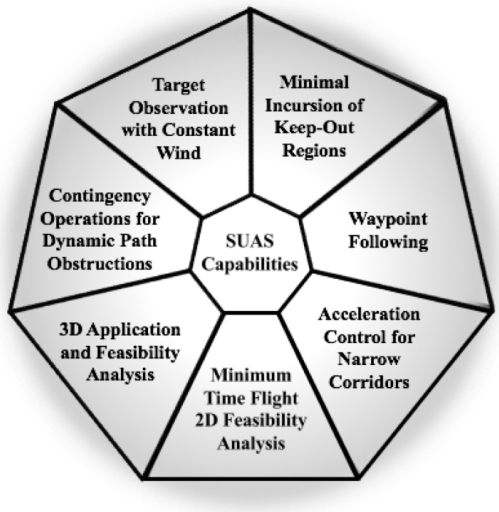

Figure 3. SUAS Capabilities

The first two scenarios evaluated analyze the feasibility for the two and three-

dimensional case, followed by waypoint tracking through narrow constraint corridors, inclusion of keep-out regions, impact of a constant wind field, and contingency maneuvers. Each of these simulations will be designed to provide feasible solutions that are accurate and computationally efficient and demonstrate the SUAS capabilities.

1.5 Expected Contributions

The following specific contributions to the optimal control field are expected at the completion of this work.

1. A defined methodology for solving highly constrained optimal control problems for SUAS path planning by eliminating all hard keep-out constraints from the NLPs domain and reducing problem specific parameter bounds to aircraft specifications.

2. A demonstration for handling exogenous inputs to the aircraft system, to include constant wind fields and dynamic constraints. Wind fields will be incorporated within the simplex construct while contingency maneuvers will be developed for continuous operations should a CSC become obstructed.

3. Provide a foundation for optimal path solutions in three-dimensions by performing a tetrahedron discretization of the domain, acquiring a feasible simplex corridor, and implementing the methodology into the optimal control solver to acquire optimal path solutions.

1.6 Document Outline

This dissertation document contains six chapters and six appendices. The first chapter introduces the motivation, defines the research questions, scope, and tasks,

sets a baseline for the assumptions and limitations, and outlines the expected contributions. Figure 4 describes the topology of the work.

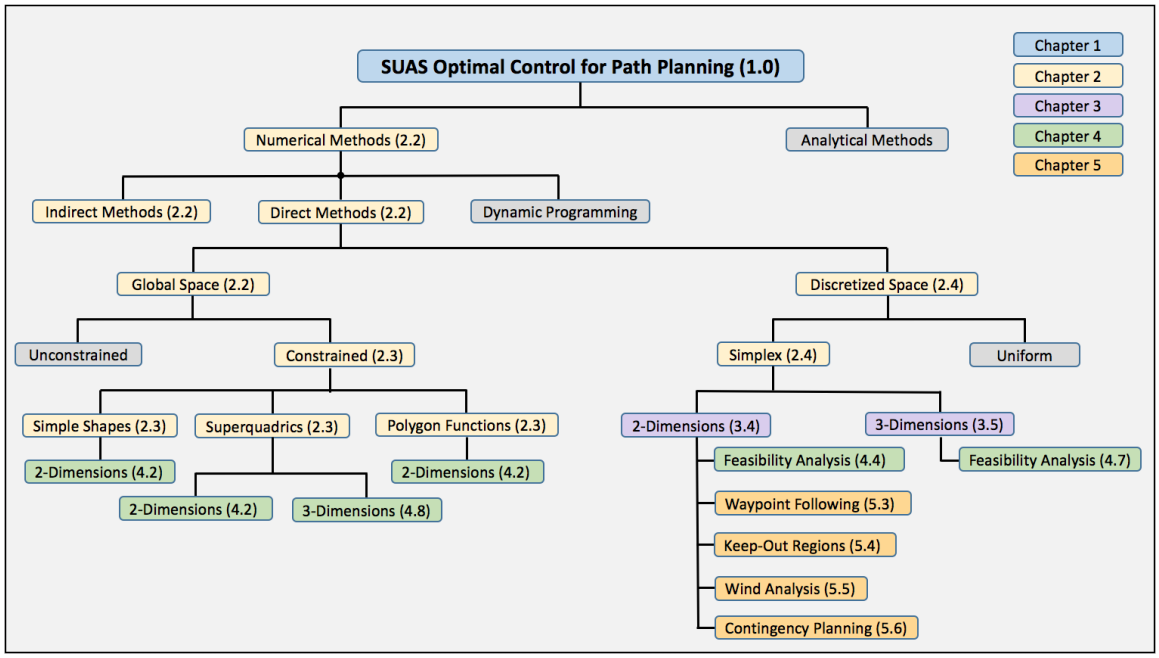

Figure 4. SUAS Optimal Control Topology & Chapter Layout

Chapter II provides an open literature review on previous methods and solution techniques for solving constrained optimization problems. Chapter III describes the methodology and techniques used for this research effort and defines the optimal control problem for the two-dimensional and three-dimensional problems. Chapter IV presents scenarios for constraint models, the two-dimensional problem through a constrained map of downtown Chicago, USA, and a feasibility analysis extending the two-dimensional problem to three dimensions. Chapter V presents four additional scenarios providing variations to the required mission or utilized cost function. This includes waypoint tracking, minimizing incursion to keep-out regions, a constant wind analysis within the simplex construct, and contingency operations should flight path corridor become obstructed. Finally Chapter VI provides a conclusion for the work and a recommendation for future work within the CSC construct.

II. Literature Review

2.1 Introduction

The optimal control problem for determining optimal flight trajectories for SUAS path planning is one that presents many challenges. This chapter provides a literature review covering three main topics in optimal control design for SUAS. The first section addresses optimal control theory while examining the differences between indirect and direct solution methods, the impact of modeling constraints on the search space, and the effects the quality of an initial guess has on the solution. The second section explores computer animation techniques and comes to an understanding of how search algorithms are chosen and implemented to traverse autonomous agents through a field of hard path constraints, often implemented within computer animation. Finally, the third section examines techniques that may be able to bridge the gap to real-time solutions by combining the fast convergence times of the geometric path planning algorithms with the high fidelity solutions for complex dynamics associated with optimal path planning for SUAS.

2.2 Optimal Control

Dating back to the mid 1980's, direct collocation methods have been used to solve the optimal control problem for unmanned vehicles. At the Air Force Institute of Technology (AFIT), the application has covered a range of topics, including collision avoidance [12], loyal wingman [13], hypersonics [14], missile avoidance [15], Air Force range flight safety [16], and many more in the the aerospace field. In each of these examples, computation speeds and convergence issues have prevented the algorithms from being used for onboard, real-time operations. Optimal control solvers such as GPOPS-II have increased the accuracy and computation speed for many optimal

control problems, however there remains convergence issues that can arise and prevent a solution from being determined. First, any gradient-based optimal control solver requires an initial guess to the solution. This guess can have a great affect on the optimal solution found, which may or may not be the global optimum. If a poor guess is presented, the NLP solver may converge to a local minimum rather than a global minimum, or worse, convergence may not be achieved and therefore a solution may not be found. Second, constraints can be difficult to model since the NLP solver requires that the obstacle be represented with a continuous and differentiable function. Because of this, most obstacles described in the literature use circular or elliptical shape functions to represent an obstacle. However, many constraints such as buildings or mountains, have straight line edges which become more problematic to represent without introducing error into the shape model. Third, using collocation points can create a situation where the optimal path indirectly violates a hard constraint. If a hard constraint is long and narrow or has a sharp corner, a solution using collocation points could pass directly through the constraint, or cut the corner of the constraint without adding a penalty to the objective function but clearly violating the constraint. Finally, an optimal control solver requires the problem to be bounded in the domain of the states, control, and time. This can be challenging for many problems as there is uncertainty as to how large or how tight to make these bounds which can be directly related to the computation time required for solving the problem.

One reason programs such as GPOPS-II are successful is because they take advantage of a sparse Jacobian matrix, built through quadrature and consisting of the system dynamics, path constraints and boundary constraints [17]. Computation time can be increased as the data required in the Jacobian matrix is minimized. By eliminating constraints from the problem formulation, less data is required in the Jacobian matrix and convergence times can be improved. These four issues have all contributed

to the challenge of developing a robust, real-time optimal control solution.

The Optimal Control Problem.

The optimal control problem solutions satisfy differential equations subject to a number of boundary and path constraints while minimizing (or maximizing) a performance index. Due to the complexity of most optimal control problems, analytical solutions cannot be obtained and therefore numerical methods are employed to find a feasible solution. The two common methods studied in the literature to solve the optimal control problem are indirect and direct methods. Both methods have been documented extensively [18, 12, 19] and therefore only an overview will be covered in this section. Each method begins by defining a set of differential equations that describe the dynamics of the system as a function of the state variables, x, the control, u, and the time, t.

$$\dot{x} = f(x(t), u(t), t) \tag{2.1}$$

The objective is to find an admissible control, u^* that minimizes the performance index defined as a summation of the terminal cost, Φ, plus the integrated cost, $\mathcal{L}$, described in Bolza form [20] as

$$J = \Phi(x(t_o), t_0, x(t_f), t_f) + \int_{t_0}^{t_f} \mathcal{L}(x(t), u(t), t)dt. \tag{2.2}$$

The objective function can be formulated to solve a multitude of problems. The one that will be used predominantly in this work, and the most common, is simply minimizing the time for the dynamics to propagate from the initial condition to the

terminal condition, describes as

$$J = (t_f - t_0) = \int_{t_0}^{t_f} 1dt. \tag{2.3}$$

A second example is to minimize the control effort required to propagate the dynamics,

$$J = \frac{1}{2}\int_{t_0}^{t_f} u(t)^T R u(t) dt \tag{2.4}$$

where R defines a positive-definite weighting matrix applied to the control vector. A third example again minimizes the control effort while also minimizing the deviation from a known or desired path, C,

$$J = \frac{1}{2}\int_{t_0}^{t_f} [(x(t) - C(t))^T Q (x(t) - C(t)) + u(t)^T R u(t)] dt. \tag{2.5}$$

Once again, R is a positive-definite weighting matrix applied to the control, Q is defined as a positive semi-definite weighting matrix applied to the deviation of the states from the desired path. The fourth example illustrates a cost function designed to maintain a location near the origin with minimal control effort,

$$J = \frac{1}{2}\int_{t_0}^{t_f} [x(t)^T Q x(t) + u(t)^T R u(t)] dt. \tag{2.6}$$

Finally, the last cost function example incorporates the ϕ term by minimizes the distance to a final location of the state with minimal control effort,

$$J = \frac{1}{2}(x(t_f) - x_f)^T S_f (x(t_f) - x_f) + \frac{1}{2}\int_{t_0}^{t_f} u(t)^T R u(t) dt, \tag{2.7}$$

where S_f defines a positive semi-definite matrix applied to the final state and x_f defines the desired final location [21]. To define the boundary of the problem, constraints are added for the initial and terminal states

$$\psi(x(t_0), t_0, x(t_f), t_f) = 0; \tag{2.8}$$

while inequality path constraints may be included to eliminate regions of the solution domain

$$C_U \leq g(x(t), u(t)) \leq C_L. \tag{2.9}$$

Indirect Methods.

Indirect methods apply the calculus of variation to transform the optimal control problem into a Hamiltonian Boundary-Value Problem (HBVP). The Hamiltonian relates the optimal states and controls to the optimal co-states, $p(t)$ described as

$$\mathcal{H}(x(t), u(t), p(t), t) = \mathcal{L}(x(t), u(t), t) + p^T(t) f(x(t), u(t), t). \tag{2.10}$$

By differentiating the Hamiltonian with respect to the states, controls, and co-states, the first-order necessary conditions for optimality are formed [20].

$$\dot{x}^*(t) = \frac{\partial \mathcal{H}}{\partial p}(x^*(t), u^*(t), p^*(t), t) \tag{2.11}$$

$$\dot{p}^*(t) = -\frac{\partial \mathcal{H}}{\partial x}(x^*(t), u^*(t), p^*(t), t) \tag{2.12}$$

$$0 = \frac{\partial \mathcal{H}}{\partial u}(x^*(t), u^*(t), p^*(t), t) \tag{2.13}$$

Solving the first-order necessary conditions results in a set of extremal trajectories. The optimal control, u^* is chosen as the trajectory which produces the lowest value

evaluated by the performance index. Equation (2.11) and (2.12) relates the states to the co-states while Equation (2.13) requires that the optimal control minimizes the Hamiltonian. If constraints are included in the problem setup, slack variables are used to translate the constrained problem to an unconstrained problem. In solving the optimal control problem with the indirect method, the co-states are solved first and the optimal control is determined through the co-states, in other words the optimal control is determined "indirectly". This method produces a highly accurate solution and assures the first-order optimality conditions are satisfied.

Disadvantages to the indirect method include a small radii of convergence and the requirement to analytically derive the HBVP. A good guess for the states, control and co-states is required for convergence, which often becomes time consuming and problematic, specifically for the co-states which have no obvious physical meaning to the problem [22].

Direct Methods.

Many times the optimality conditions are difficult to formulate and determining a realistic estimate for the co-states is not intuitive. To avoid these issues, direct methods transcribe the infinite-dimensional optimal control problem into a finite optimal control problem with algebraic constraints, otherwise known as an NLP [23]. There have been many methods developed to transcribe the optimal control problem into an NLP, including direct shooting methods [24], state and control parameterization methods [25], and direct orthogonal collocation methods [12, 18]. The focus of this research will be on direct orthogonal collocation methods, also referred to as psuedospectral methods in the Aerospace field of study [23].

When solving an optimal control problem with direct orthogonal collocation, the continuous time optimal control problem is transcribed to a discretized nonlinear

programming problem. This is accomplished with three main concepts; orthogonal collocation, polynomial approximation, and Guassian quadrature [19]. For this research, the continuous functions of the optimal control problem are approximated with a finite dimensional Lagrange polynomial basis [26]. The state, x, is approximated at a set of collocation points described as

$$x(\tau) \approx \tilde{x}_N(\tau) = \sum_{i=1}^{n+1} x_i L_i(\tau) \tag{2.14}$$

where $\tilde{x}_N$ is the N point approximation of $x(\tau)$, x_i represents the weighting function, $L_i(\tau)$ is the Lagrange polynomial basis

$$L_i(\tau) = \prod_{\substack{i=0 \\ j \neq i}}^{n} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{2.15}$$

and τ represents an affine transformation of the time t on the interval from $(-1, 1)$ by

$$\tau = \frac{2t - (t_f + t_0)}{t_f - t_0}. \tag{2.16}$$

For this research Legendre-Gauss-Radau points will be implemented [19]. With the problem discretized, Gaussian quadrature is used for differentiation or integration of the state and control. This method is termed a global method as each collocation point is solved simultaneously rather than other fixed interval methods such as a three or five point formula method [27]. The error produced by this method can be greatly reduced by choosing the collocation points appropriately with Legendre or Chebyshev point placement to minimize the affects of Runge phenomenon.

One disadvantage of the direct method results from the discretization of the op-

timal control problem producing several minima, leading to a solution that may be far from the global optimum. To minimize this affect, an accurate prediction of the state, control, and time are required to seed the NLP. The quality of the prediction will have a direct affect on the feasibility of the solution, as there is no guarantee of convergence to a global minima with direct methods. Many algorithms have been proposed previously to produce an initial guess to the solution, including Dubins path algorithms [28] and heuristics [29, 30] with computation time and accuracy being the limiting factor for complete hybrid solutions.

2.3 Constrained Trajectory Optimization

Direct orthogonal collocation has quickly become a popular method for determining optimal trajectories of air breathing vehicles with mission constraints. Boundary constraints are often placed on the geographical location for the aircraft's starting point and terminal point. The vehicle dynamics may contain constraints such as altitude limitations and the vehicle control may be restricted by limiting factors such as the max turning rate, velocity, or angle of attack. Path constraints can be imposed to represent no-fly zones, infrastructure, or terrain. Each of these constraints are mission specific and are required to be updated for each flight profile.

The complexity of the optimal control problem can grow exponentially when constraint models are incorporated. These models can range from simplistic shapes representing circular or elliptical regions, to superquadrics, or even polygonal shapes [28]. Each of these must be modeled as a path constraint in the optimal control problem, reducing the sparsity of the Jacobian matrix and increasing the computational requirements. Further, these path constraints must be smooth differentiable functions in order to quickly acquire the Jacobian and Hessian. This can be problematic when designing algorithms to handle multiple constraints in a computationally

efficient fashion.

Basic Shapes.

Many techniques have been employed to appropriately model path constraints in two-dimensional space. One of the most commonly employed techniques is to simulate a vehicle no-fly zone with a circular or spherical constraint function [31, 18, 14] of the form

$$\|x_k(t) - \tilde{x}_n\|_2 > r_n. \tag{2.17}$$

Here, the difference between the position states of the k^{th} vehicle, x_k and the center of the n^{th} no-fly zone, $\tilde{x}_n$, must be greater than a defined radius of the n^{th} zone, r_n. These functions are differentiable and smooth and therefore can be handled with relative ease using an NLP solver such as Interior Point Optimizer (IPOPT) or Sparse Nonlinear Optimizer (SNOPT). The drawback is that the constraint is restricted to circular or spherical shapes. This may be adequate to represent a no-fly zone, however error will be introduced for polygonal shapes representing city infrastructure or landscape that require a more polygonal shaped constraint models.

Superquadrics.

Superquadrics have been used extensively in computer vision, computer graphics, and robotics [32] and are commonly used to transition circular and elliptical shaped constraint models to polygonal shapes. For two-dimensional shapes, the superellipse centered at $(0, 0)$ is a type of Leme curve which is defined as

$$F(x, y) = \left(\frac{x}{a}\right)^m + \left(\frac{y}{b}\right)^m \tag{2.18}$$

where any point on the SUAS trajectory is outside of the constraint when $F > 1$. Variables a and b represent the semi-major and semi-minor axes of the superellipse and m is a rational number defining the shape at the corners of the object as shown in Figure 5.

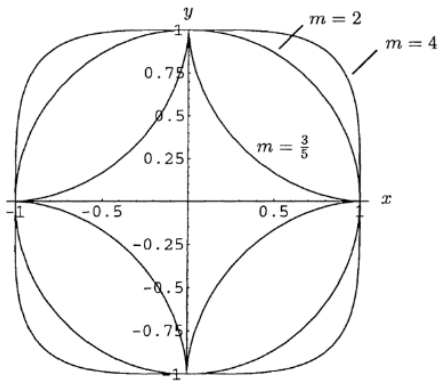

Figure 5. A superellipse can change continuously from a star-shape through a circle to a square shape in the limit (m $\rightarrow \infty$) [32]

When solving problems with direct orthogonal collocation, the superquadric has been used to include constraint shapes other than simple circular or elliptical regions. Hurni, Lewis and Mohan used the superquadric in a constraint rich environment and found success in convergence but at the cost of computation time [33, 34, 35]. Their research has shown a sufficient number of shapes can be produced to adequately represent an optimal control problem as shown in Figure 6.

The shapes, produced from Equation 2.18 with $m = p$, can be formulated into an inequality constraint referred to as an inside-outside function as follows,

$$F(x(t), y(t)) = \left(\frac{x(t) - x_c}{a}\right)^p + \left(\frac{y(t) - y_c}{b}\right)^p - 1 > 0. \tag{2.19}$$

Here, the position of the vehicle, $(x(t),y(t))$ is evaluated in the function at each collocation point to assure the vehicle remains outside the constraint located at (x_c, y_c). The NLP solver, whether it is an IP solver or a SQP solver, requires that the function

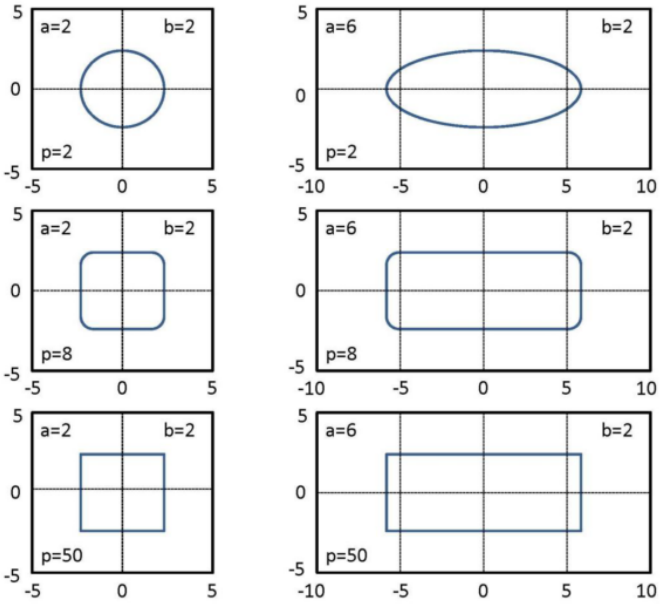

Figure 6. Various Shapes with Varying Parameters of Equation [35]

used to model constraints be continuous and smooth. As the p parameter is increased, the gradient of the function becomes excessively large at the corners and the function itself can grow without bound resulting in a poorly scaled function. To minimize the impacts this can have on an NLP solver, two common approaches are taken. The natural log can be taken of both sides of the equation to scale the path constraint and balance out the problem formulation [35]. Alternatively, the function can be incorporated into a modified inside-outside function through a sigmoid function [36]. This method allows for a bounded, continuous, and differentiable function,

$$\phi(F) = \frac{1}{1 + e^{(s(F(x,y)-1))}}. \tag{2.20}$$

Here, s represents the stiffness parameter of the curve and F as defined in Equation 2.19. Figure 7 shows a straight line vehicle trajectory, annotated with the blue asterisks, through a constrained circular region located at $(1, 1)$. The gray curves depict

the stiffness of the function as s is varied from 0.1 to 10.

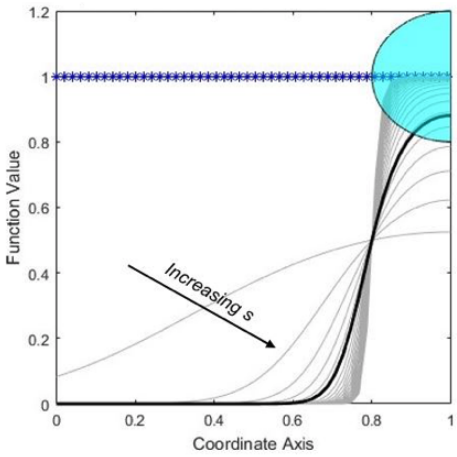

Figure 7. Stiffness Transition of Sigmoid Function

Incorporating this function as a path constraint, any functional value greater than zero represents a position close to or inside the keep-out region dependent on the stiffness parameter. The normalization of this function value can be handled through the distributed weight values when included in a cost function. Smith utilized sigmoid functions to model inequality constraints in the optimal control problem designed to minimize deviation from a flight path while maintaining horizontal or vertical separation from intruder aircraft [12]. Smith's work defined a superquatric superellipsoid around the primary aircraft, representing a safety buffer for intruding vehicles. The work presented herein applies sigmoid functions in the two-dimensional scenario to model aircraft keep-out regions and implements a stiffness value of $s = 2$, illustrated by the solid black line in Figure 7.

The formulation in Equation 2.20 has been shown to be a feasible solution to modeling two-dimensional constraints, however computation times have exceeded the required values to run the algorithm for real-time operations. Hurni illustrates this fact in his dissertation and increases computation speed by providing a previous solutions guess to the NLP solver through a bootstrap method [34]. Although this

is a common practice, the objective of this research is to solve both the initial guess and the optimal path for real-time operations onboard the air vehicle.

In three-dimensions, superquadrics have been used in numerous applications to model city streets or design aircraft components [32]. Smith used superquadric modeling to represent odd-shaped probability regions for aircraft collision avoidance [12]. Similar to the two-dimension discussion, the rounded edges of a shape can be minimized and different shapes can be constructed by altering parameters in the basic equation centered at $(0, 0, 0)$ [37]

$$\left(\left(\frac{x}{a_1}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{a_2}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{a_3}\right)^{\frac{2}{\epsilon_1}} = 1 \tag{2.21}$$

where constants a_1, a_2, and a_3 set the widths and height of the superellipsoid and ϵ_1 and ϵ_2 vary the cross-section parallel and perpendicular to the x, y plane respectively. Although superquadrics can begin to represent polygonal shapes, the additional computation time and inconsistencies that result are not practible for onboard SUAS computations.

Polygon Functions.

To further extend the constraint model, the ray-casting algorithm [38, 39], used to determine if a point is inside an arbitrary polygon, returns a boolean variable, in this case with `True` corresponding to a point outside a polygon, and `False` for a point inside a polygon. To eliminate the discontinuity along the edges of the polygons, the result of the ray-casting algorithm is then multiplied by the distance to the nearest polygon.

Given a line, denoted as $\mathbf{v}$, passing through the points (x_1, y_1) and (x_2, y_2), and an arbitrary point denoted as $\mathbf{p} = (x_p, y_p)$, let $\mathbf{n} = (x_n, y_n)$ denote the point on line

v nearest to the point **p**. The point **n** is computed as below.

$$u = \frac{(x_p - x_1)(x_2 - x_1) + (y_p - y_1)(y_2 - y_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.22}$$

$$x_n = x_1 + u(x_2 - x_1) \tag{2.23}$$

$$y_n = y_1 + u(y_2 - y_1) \tag{2.24}$$

If **v** is a line segment, $u \in [0, 1]$. Therefore if $u < 0$ according to Equation (2.22), $u = 0$ and if $u > 1$, $u = 1$. The distance from **p** to **n** then gives the minimum distance from **p** to **v**.

To compute the shortest distance to any edge in the environment, the above procedure for finding the minimum distance to a segment is computed for all segments of all polygons in the environment. The minimum distance for the environment, d_{env}, is set to an arbitrarily high value. Then, for each segment, the distance d_i is computed. If $d_i < d_{\text{env}}$, then $d_{\text{env}} = d_i$. Since d_{env} only computes the minimum distance to an edge of a polygon in the environment, it is then multiplied by the ray-casting result, computed in MATLAB® using the `inpolygon` function. The result returns a value of $d_{\text{env}} = 0$ for all points that are inside a polygon, and $d_{\text{env}} > 0$ for all points outside a polygon.

Heuristic Methods.

Thus far, only indirect and direct methods have been discussed for optimal SUAS trajectory path development. Another widely accepted method for numerical optimal control is the use of heuristics [40]. Heuristics methods are initiated with a set of possible solutions and use a stochastic search to adapt and iterate the initial solution until an optimal solution is found [29]. Heuristic methods are search based methods where the gradient of the problem is not used, therefore not requiring the problem to

be differentiable as in the direct method. These methods provide two main advantages over indirect and direct methods; the initial solutions are randomly determined and therefore no initial guess of the solution is required and the stochastic search leads to a global technique in which a global minimum is often found. However the solution must be parameterized to a low number of variables, and may not be as accurate as indirect or direct methods. Additionally, there is still no guarantee that the solver will converge to the global minimum solution. Therefore, often these two methods are used together, where the heuristic algorithm is used to generate an initial guess for the gradient-based direct method [41]. In computer animation and robotics, heuristic search techniques are used to find feasible solutions quickly through static, re-planning, and anytime algorithms. Ferguson discusses the strengths and weakness of these algorithms [30].

Each of these methods are based on the premise that the constraint is contained inside the search space and the NLP solver must evaluate constraint equations to assure a feasible solution can be attained. The following section leverages research from the field of computer animation to eliminate the constraints from the search space by performing a triangular discretization dependent on the constraint field rather than the size of the search space.

2.4 Fast Geometric Path Planning

Direct orthogonal collocation methods are capable of meeting the computational requirements for onboard SUAS operations if an adequate starting point of the states are given to the NLP solver. When approaching the constant altitude SUAS problem, comparisons can be made to path development of autonomous agents in the interactive virtual world. Virtual worlds are populated with autonomous virtual humans, or agents, where computed paths must take into account path length, time, and energy

expended traversing the path [42].

The demand on the complexity of the virtual world combined with improved graphic capability has encouraged new techniques for achieving intelligent navigation for the next generation virtual agent simulation [43]. Given the speeds of which these video games are played, path planning algorithms must perform efficiently under limited time budgets to traverse the autonomous agent from one end of a building to the other. Often greedy algorithms are chosen in which the first feasible path that is found is continuously improved according to the characteristic dynamics and a pre-determined threshold on computational time. The path available at the time required is then implemented [44, 42]. This can often result in a sub-optimal path, but in the virtual world of the gaming industry, the timeliness of the solution is more heavily favored over the most optimal route.

2.5 Discretization Methods

Grid representations are a discretization of the space that allow for strict guarantees on optimality and the completeness of a solution. One application is related to computer animation and the path planning of autonomous agents [45]. The simplest form of discretization is a uniform square grid. The performance and quality of the solution are heavily dependent on the resolution of the grid, with fine resolutions leading to increased computation time that are not feasible for real-time operations [42]. Other methods include restricting path selections to specific roadmaps [46], cell and portal graphs where the cells represent small search spaces and portals represent doorways to other cells using a generalized Voronoi diagram [47], and simplex search spaces where the domain is divided into triangular regions allowing for avoidance of obstacles [48]. As each discretization method has advantages and disadvantages, this research will focus on a simplex discretization method called Delaunay triangulation.

Delaunay Triangulation.

Triangulations have been used for the purpose of computer navigation queries due to their powerful representation of the structured environment. One of the main advantages of triangulation over a uniform grid is that the triangular decomposition of the space has $O(n)$ cells, where n is the number of segments used to describe the obstacles in the environment [49]. This allows for a discretization technique that focuses on the complexity of the environment rather than the size of the space. Delaunay triangulation is formed through the Dirichlet tessellation which is described in detail in [50] and [51]. The result produces a set of triangles where no data point will lie inside the circumcircle of any formed triangle. As a simple example, five points are illustrated in Figure 8 below. The Dirichlet tessellation is synonymous with the Voronoi diagram [52] and is shown with the dotted lines. Here the vertices of the tessellation is equidistant to the corresponding triangular vertices. The Delaunay triangulation is formed with perpendicular segments to the tessellation lines and is illustrated with solid lines [51].

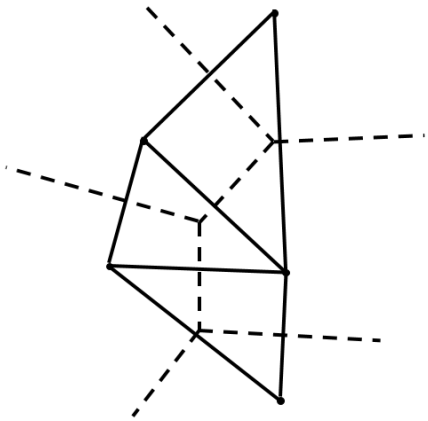

Figure 8. The Dirichlet Tessellation and Delaunay Triangulation [51]

The formal definition of a Delaunay triangulation can be found in [53] and is

described as follows:

Definition. Let S be a set of points in the plane. A triangulation T is a Delaunay triangulation of S if for each edge e of T there exists a circle C with the following properties:

1. The endpoints of edge e are on the boundary of C.

2. No other vertex of S is in the interior of C.

The Delaunay triangulation may not always be unique [54]. Consider the case where you have 4 points to form a square. The tessellation point is located at the center of the square and therefore the Delaunay triangulation has two configurations, both which are valid. The first divides the square from the top left to the bottom right where the second divides the square from the top right to the bottom left. In this case, either configuration may be chosen to form the triangulation. A more troubling scenario is apparent when a constrained edge is required in the search space, forcing the triangulation to be represented into a specific set, which may violate one of the definitions given above. In this scenario, a *constrained* Delaunay triangulation must be formed.

Constrained Delaunay Triangulation.

Often, fast geometric path planning algorithms discretize the search space with a triangular mesh formed with a Constrained Delaunay Triangulation (CDT). A CDT is a refinement of the Delaunay triangulation that forces a required segment as an edge of the triangulation. The formal definition is described as follows [53]:

Definition. Let G be a straight-line constrained edge. A triangulation T is a CDT of G if each edge of G is also an edge of T and for each remaining edge e of T there exists a circle C with the following properties:

1. The endpoints of edge e are on the boundary of C.

2. If any vertex v of G is in the interior of C then it cannot be "seen" from at least one of the endpoints of e (i.e., if you draw the line segments from v to each endpoint of e then at least one of the line segments crosses an edge of G).

The CDT is made of constrained edges and unconstrained edges. When determining an optimal path, the solution can cross an unconstrained edge, but must avoid all constrained edges. Since the constrained edges are required, often formulation of the CDT contains a vertex or an edge that does not satisfy the Delaunay triangulation conditions, and therefore a CDT may not be equivalent to a Delaunay triangulation. An example of the differences can be seen below in Figure 9.

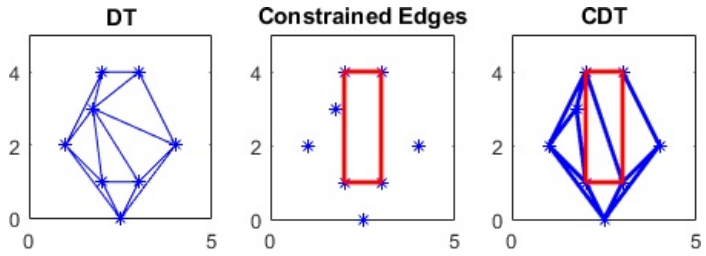

Figure 9. A: Unconstrained Delaunay Triangulation; B: Constrained Edge Segments; C: Constrained Delaunay Triangulation [53]

Here, Figure 9A shows a Delaunay triangulation without any constrained edges. Figure 9B implements constrained edges into the field and Figure 9C illustrates the CDT with the constrained edges included in the triangulation.

Using this technique, constraints can now be forced into the discretization of the space. Applying these constraints to the SUAS path planning problem, these boundaries illustrate hard constraint no-fly zones that represent polygonal constraint shapes such as buildings, terrain, and restricted airspace.

Path Search Algorithms.

Path planning algorithms have been designed to determine the best route from an initial starting point to a goal location while satisfying conditions on constraints and dynamics. In the field of optimal control, the dynamics of the agent must be satisfied while obeying all constraints placed on the states, control, and path. In computer animation, path planning focuses on determining the best route for the autonomous agent while avoiding walls, furniture and other obstacles. Many algorithms have been developed to accomplish autonomous agent path planning while placing an emphasis on computational speed as solutions must be found quickly to give the appearance the agent is in fact making real-time intelligent decisions as the video game progresses [55]. The following briefly outlines a few of the most used algorithms and the advantages and disadvantages of each.

Dijkstra's Search Algorithm.

Dijkstra's search algorithm is an iterative procedure that repeatedly attempts to improve an initial approximation of the path cost. It solves the dynamic programming functional equation by the reaching method [56]. The algorithm is initiated with a discretized space forming a collection of nodes defined by integers 1 through N. The initial node, designated by the starting point, is given a function value $f(1) = 0$. The first iteration than evaluates each connecting node based on the formula

$$v_j = min\{v_j, v_i + d_{ij}\}. \tag{2.25}$$

The node location is expanded outward for each iteration, updating the value of each node in a tree-like fashion [35]. This procedure will set a value for each node in the discretized space. Since each node was evaluated, the optimal path can be calculated

between any two nodes in the domain. This algorithm takes advantage of Bellman's Principle of Optimality, stating the "optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first transition" [57].

Dijkstra's algorithm searches the entire domain, allowing for a global optimal solution upon completion. Additionally, in a stochastic environment, if the agent is perturbed off the optimal path, the optimal path from the agent's new location has previously been calculated and can immediately be implemented. A consequence of this algorithm is that the entire space must be searched, adding computation time that the video game industry cannot afford.

Myers et al. executed the Dijkstra search algorithm to find minimum paths for UAVs through polygon obstacles [58]. Their research focused on the two-dimensional search with basic vehicle dynamics in a node-to-node search resulting in a Dubins path model. They cite difficulties guaranteeing feasible paths while minimizing computational time and their algorithm only allows finite increments in the vehicle control for heading with a constant speed. The Dijkstra search algorithm implemented in Myers' research does not account for flight dynamics, and therefore a scenario could unfold where the CSC through a constraint field is infeasible due to the flight limitations (minimum turn radius) of the vehicle.

A* Search Algorithm.

The A* search algorithm gets its roots from Dynamic Programming and is an extension to Dijkstra's algorithm. Whereas the entire domain is searched with Dijkstra's algorithm, an A* search is guided by a heuristic. It is considered an "informed search" or "best-first search" as the algorithm searches the space until a solution is found. Once a solution is found, the search continues to look for additional solutions

with a lower cost value. If a lower cost is discovered, that solution replaces the previous solution. This process continues until the entire space has been searched or a computational time limit has expired. If no time limits are imposed, the result of the algorithm will be consistent with Dijkstra's algorithm. However, in order to find a solution in the quickest possible manner, time limitations are often placed on the algorithm which may result in a sub-optimal path solution. In computer animation, this is a trade-off in order to achieve the computational times required for video game processes.

The A* algorithm is initiated in the same manner as the Dijkstra algorithm. The initial node is given a cost value of $f(1) = 0$ and additional nodes are searched in a tree like fashion. A heuristic is introduced and added to the cost function to give a priority in the direction of the search. Often the Euclidean distance is used as the heuristic and the overall cost function is

$$f(n) = g(n) + h(n) \tag{2.26}$$

where $g(n)$ is the cost evaluated at the current node and $h(n)$ is a heuristic that estimates the cheapest path from the current node to the endpoint [59]. At each iteration, the A* algorithm must determine which nodes to expand by selecting the path that minimizes the current cost, $g(n)$ and the cost to go, $h(n)$. In the case where $h(n)$ is set equal to 0, the result will be consistent with Dijkstra's algorithm [60].

Disadvantages of the A* algorithm are related to the choice of heuristic which must be chosen to underestimate the cost, thus giving a lower bound to the solution. This may cause the algorithm to spend computation time discriminating between two paths of equal remaining distances. Additionally, the result produces an optimistic *estimate* of all possible solutions, resulting in a path that may be sub-optimal if the algorithm is terminated after the goal node is first reached [59].

RRT Search Algorithm.

Rapidly exploring Random Tree (RRT) algorithms first developed in the late 1990's by Lavalle [61] and were designed to search non-convex spaces of high dimension by randomly building a space-filling tree. Specifically, they were built for path planning with obstacles and differential constraints most commonly used for autonomous robots. The algorithm consists of a tree of feasible trajectories by extending branches toward randomly generated target points [62]. The search tends to advance in large regions of unexplored space. The probability of the expansion of the existing state is proportional to the size of the Voronoi regions, lending to large regions being on the cusp of the search space. A significant feature of the RRT algorithm is that it has been proven to be probabilistically complete, that is, the probability of finding a complete path converges to one if a feasible path exists [62]. An illustration of the RRT algorithm is given below in Figure 10.

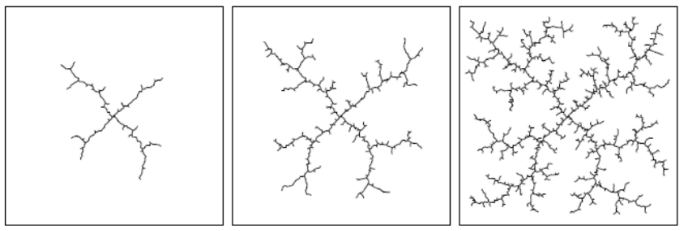

Figure 10. RRT Expansion Example; Starting from the Center of a Square [61]

The disadvantage of this algorithm results from the randomly generated target points used for the tree expansion. These points can become problematic when solving the optimal control problem as points may be defined within an arc of minimum turn radius. This creates points that require multiple turn maneuvers to achieve a desired path and add to the computational time of the algorithm.

A hybrid solution for optimal control was developed by Aoude using an RRT algorithm to develop an initial guess for an optimal control solution by pseudospectal

methods [63]. The goal of his research was to develop real-time path planning for multiple spacecraft reconfiguration maneuvers with various path constraints. His results show the benefit of using an RRT algorithm as times were drastically reduced, but did not reach a level required for onboard operations [64] and a consistency in solution convergence was not achieved.

Triplanner Toolkit.

Marcello Kallmann has provided an extensive review of path development with clearances in [65] and has developed the Triplanner toolkit algorithms designed to determine the shortest path while accounting for minimum clearance distances to all constraints [45]. Specifically, he focused on determining paths of shortest distances, efficiency of computation time, and maintaining proper clearance from obstacles along the calculated path. An overview of the development of the Triplanner algorithm is discussed below. A more extensive review of the algorithm can be found in [65, 66].

First, let S define a set of n segments that form all the constrained edges in the domain. The set of all endpoints of each segment then form the set P. A CDT, T, is then formed such that all segments of S are also segments of T and the Delaunay criterion are upheld. Next, Kallmann performs a test to assure that a disk of radius r can traverse through any given region. This enables an efficient computation of paths with arbitrary clearances. If a local clearance test fails, a refinement of the mesh is performed by redistributing the triangulation or adding a vertex point to a straight line segment of the set S. This may result in a new set of constrained edges, as a straight line segment could be subdivided into multiple sections. Once the triangulation has passed the local clearance test, the final mesh is termed a Local Clearance Triangulation (LCT).

A path through the LCT from a starting point p to a finish point q is defined

as "free" if it does not cross a constrained edge. A free path will cross several unconstrained edges resulting in a CSC formed of all traversed triangles. Using an A* algorithm in a discretized space formed by the triangulation, a geometric solution for the CSC is determined. The CSC formed by the A* search is dependent on the chosen cost function and will result in a shortest path search defined by the cost metric. A simplistic cost metric may consist of using the Euclidean distance between the centroid of each triangle or instead using the midpoint of each unconstrained segment to determine the best CSC. Kallmann settled on a cost function which begins the search from the midpoint of each triangle, but enhanced the search through a reference point connecting the point in the previously traversed edge to the final point q. If q is not visible on the straight line path, the nearest vertex is used for the traversed point while accounting for the radial offset distance form each vertex point.

After the CSC is determined, a funnel algorithm is used to find the shortest path within the CSC. The funnel algorithm was developed by Lee and Preparata and Chazelle [67, 68] as cited by [69]. The algorithm has been used to calculate the shortest path under multiple applications; including path finding for autonomous agents [70], querying visible points in large data sets to define shortest paths [71], shortest paths for tethered robots [72] and robots in extreme terrain [73].

Given a CSC with a starting position p and final point q, the funnel algorithm defines the entrance point of the simplex as the apex, a, and the starting point in the funnel. Since the CSC has already been defined using the A* algorithm, the two vertices connecting the next simplex can be defined as u and v, with the third vertex w. If the straight line solution from a to w is feasible, a straight line path is chosen from a to w as shown in Figure 11A. Maintaining a as the apex, the third vertex of the next simplex is evaluated for the straight line path solution from a to w' as shown in Figure 11B. In the case where the straight line solution is not feasible, the

vertex which is closest to the next point in the path is chosen as the new apex, a' and the algorithm continues as shown in Figure 11C. Further detailed explanation of the funnel algorithm can be found in [69]. To deal with the local clearance around obstacles, Kallmann implemented a required circle constraint of radius r on each included vertex as illustrated in Figure 11D [49].

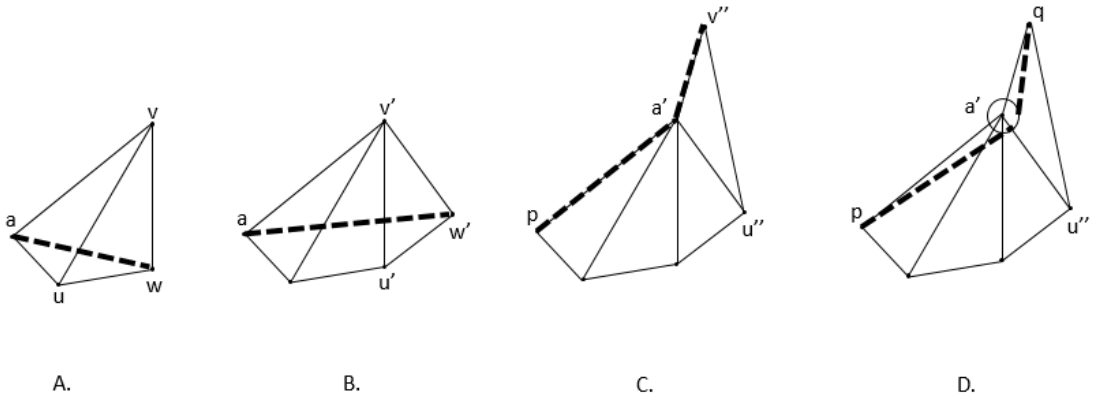

Figure 11. Implementation of the Funnel Algorithm on a Simplex Corridor

The Triplanner toolkit performs a locally optimal search and is capable of achieving path solutions on the order of milliseconds for environments with 60K+ segments [65]. Although there is no guarantee that the solution is the global optimal path, the CSC found and searched is free of constraints allowing for solutions consisting of straight line segments and minimum turn radius arcs.

Barycentric Coordinate Frame.

Often when dealing with simplex shapes, the barycentric coordinate frame is preferred over traditional Cartesian coordinates. A barycentric coordinate system defines the location of a point within a simplex as a weighted measure to each of the vertex points, also referred to as areal coordinates in the context of triangles [74]. The barycentric coordinate system defines the side of the simplex as the axes, allowing for

simple representations of lines, points, and perpendicular relationships [75].

To define the coordinate system, let r_1, r_2, ..., r_n be n vertices of a complex planar polygon Q. For the purposes of this research, when investigating problems in $\mathbb{R}^2$, triangulation techniques will be used and therefore $n = 3$. Solving problems in $\mathbb{R}^3$, tetrahedral techniques will be used and therefore $n = 4$. Any point, R, inside polygon Q can be represented in barycentric coordinates defined with the vertices of Q used as a basis as follows [76, 77, 78]:

$$\mathbf{R} = \sum_{j=1}^{n} \alpha_j \mathbf{r_j} \tag{2.27}$$

where α represents the barycentric weights, defined as a set of real coefficients whose added sum equals unity,

$$\sum_{j=1}^{n} \alpha_j = 1. \tag{2.28}$$

To ensure that each point remains inside the polygon, Q,

$$0 \leq \alpha_j \leq 1 \ \ \forall j \in [1...n]. \tag{2.29}$$

For the triangular relationship, $n = 3$, transformation from a barycentric coordinate frame to a Cartesian coordinate frame can be easily performed through a linear transformation of the coordinates represented as:

$$\mathbf{R} = \mathbf{QA} \tag{2.30}$$

where $\mathbf{R} \in \mathbb{R}^2$ defines the point location in the Cartesian coordinate frame, $\mathbf{Q} \in \mathbb{R}^{2\times n}$ represents the matrix of vertices in the polygon, and $\mathbf{A} \in \mathbb{R}^{\mathbf{n}}$ describes the weighting matrix and the set of barycentric coordinates.

2.6 Summary

Solutions to optimal flight trajectories for SUAS in constrained environments are not new to the field of aeronautical optimal control. Constraints have been modeled with several known techniques which each have their individual drawbacks. Even with the development of direct orthogonal collocation methods, convergence times for these problems have not approached the speeds required for real-time onboard operations and there are no guarantees a simulation will converge under varying initial conditions and dynamic constraints. The two biggest factors contributing to this problem are the constraint models and the initial guess provided to the NLP solver.

The computer animation industry has developed a proven method for the two-dimensional problem to quickly acquire feasible path solutions. It is proposed in this research that by using triangulation techniques and heuristic search algorithms, constraints can be eliminated from the problem formulation and a hybrid method can be developed in which an initial guess can be acquired through CSC methods and provided to the optimal control solver. In this way, through a phased approach, feasible flight path trajectories that are suitable to the TOBS SUAS problem can be acquired efficiently and accurately providing fast, reliable solutions to the constrained optimal control flight path planning problem.

III. Methodology

This chapter presents the methodology used to evaluate the problem presented in Chapter I. The aircraft model is presented followed by an overview of the software developed and used in this research. Next, the general optimal control problem and its implementation into the optimal control software GPOPS-II is described. Finally, the Triplanner toolkit algorithms are identified for use as the initial guess to the NLP solver for the two-dimensional problem while new CSC methods are developed to illustrate the feasibility of the three-dimensional problem.

3.1 Overview

To properly understand the optimal path solution of a SUAS, first the aircraft system dynamics must be established, a solution tool must be chosen, and the parameters describing the domain, constraints, and limitations of the problem must be defined. The required effort in properly setting these parameters depends on the solution method chosen to solve the optimal path.

The objective of this research is to develop a solution method that produces a feasible path while minimizing computation time to allow for real-time operations for the TOBS mission. Oftentimes in optimal control solutions, the algorithms used are designed to find the global optimal path. This can result in solutions that require problem specific input parameters and an initial guess for the solution that is very close to the optimal solution, both of which can be computationally expensive to acquire. Further, even with proper input parameters, there is no guarantee the algorithm will converge to a feasible solution.

3.2 Software

Multiple software packages are used to acquire solutions to the optimal control problem throughout this research. The primary tool is the MATLAB® based General Purpose Pseudospectral Optimal Control Software-II (GPOPS-II). GPOPS-II is used to solve multi-phase optimal control problems with a variable-order Gaussian quadrature collocation method [79]. The program translates the continuous optimal control problem into a sparse NLP. Based on a user defined accuracy, a mesh refinement method is performed to determine the number of mesh intervals and the degree of the approximating polynomial [80]. GPOPS-II is implemented in this research with an hp-adaptive version of the Radau pseudospectral method. This method evaluates the midpoint of each initially defined segment against the dynamics and path constraints to determine if the collocation points should be increased or the mesh should be subdivided before a solution is returned [17]. For each simulation in this research, the GPOPS-II settings remain consistent with only adjustments to the mesh refinement tolerance. The standard settings include the derivative approximation set to either "SparceCD" or "adigator", the mesh method set to"hp-PattersonRao", and the NLP error tolerance set to 10^{-5}.

All simulations are conducted with GPOPS-II in MATLAB® version R2016b. All results were solved on a 2016 iMac with 2.8GHz quad-core Intel i5 processor and 8GB of 1867 MHz LPDDR3 memory.

The Triplanner toolkit developed by Kallmann was delivered to the Power and Controls Division at the Air Force Research Laboratory (AFRL) in a collaboration effort to extend the algorithm principles to SUAS control and optimization theory. The Triplanner toolkit accepts constraint vertices as an input to the algorithm and performs both an A* and funnel algorithm to search for a locally optimal solution with a minimum clearance distance to any constraint for the two-dimensional problem.

The program is written in C++ and compiled to run on a linux operating system. AFRL has written a wrapper function in Python to call the Triplanner toolkit and provide desired constraint vertices as an input while returning the CSC and path solution as an output.

3.3 Triplanner Toolkit

Fast geometric path planning algorithms, such as the Triplanner toolkit, have been designed to formulate path solutions in constrained, two-dimensional environments on the order of milliseconds. The input for the search algorithm requires the vertex of each constraint in the environment and the initial and final position. Figure 12A shows building constraints as red polygons and a green and red asterisk respectively for the initial and final position. A CDT is performed on the space and is refined with a LCT in order to maximize potential simplex corridors. The simplex discretization is shown in Figure 12B. Figure 12C evaluates the feasible space by including a constant radius circle placed on each vertex point to account for the minimum turning radius of the aircraft. Finally, an A* search algorithm is executed to determine a set of simplexes that form a feasible CSC from the starting point to the final location. This path defines a CSC that is free of constraints. The autonomous agent's path is determined with a funnel algorithm which excludes the interior of the constant radius circles, resulting in a feasible flight path through the defined CSC. This assures that there is enough clearance for the autonomous vehicle through any chosen CSC. The path resembles a Dubins path solution as a series of straight line segments connecting the tangent points of the appropriate circles. An example of this CSC and resulting Dubins path is shown in Figure 12D.

The aircraft dynamics are accounted for by a radial offset distance from each constrained edge equal to the SUAS minimum turn radius. The heading angle is

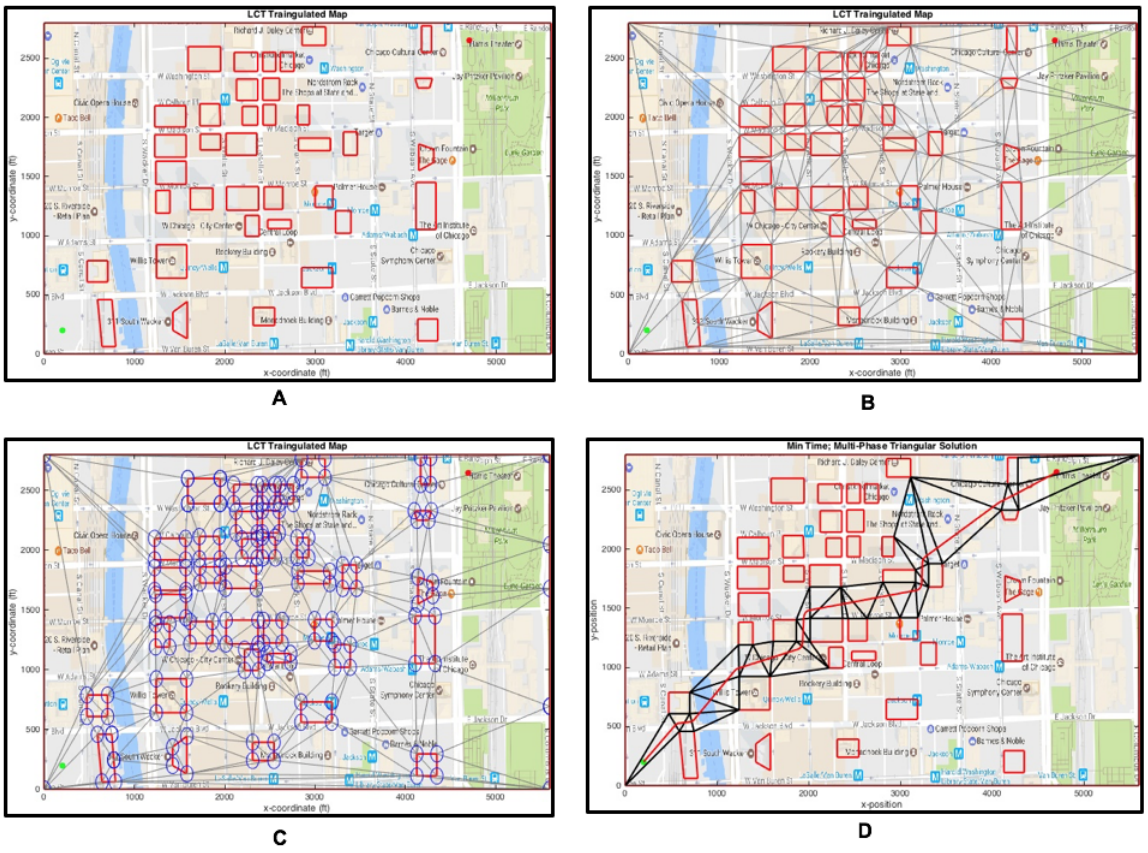

Figure 12. Implementation of the Triplanner Toolkit. [Map data @2017 Google]

estimated as the angle difference between consecutive points on the path, while the angle rates are determined with a three-point finite differencing method of the heading angle vector. The velocity vector is estimated with a maximum value during straight portions of the Dubins path and minimum values on the minimum radius turns. This path solution will be used as an initial guess for the two-dimensional SUAS optimal control problem in GPOPS-II to acquire the optimal path through the defined CSC.

3.4 Tetrahedron Discretization

Several challenges are presented when expanding the two-dimensional problem to three-dimensional space. First, the Triplanner toolkit, used to discretize the two-

dimensional space and provide an initial Dubins path solution, is not capable of expanding to the third dimension in its current state. To apply the same concepts for discretizing the space, a simple constraint map, shown in Figure 13, is constructed so that a feasibility analysis can be performed and demonstrated.

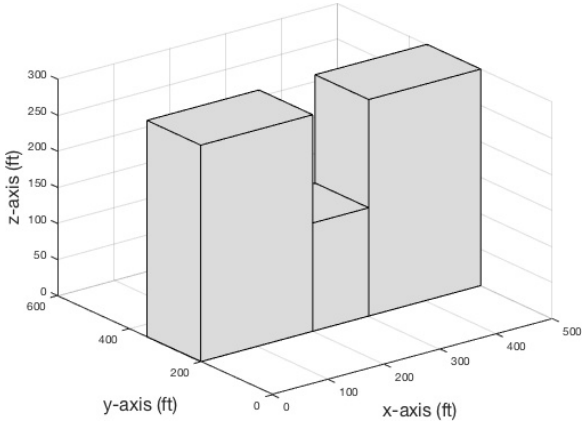

Figure 13. Three Dimensional Constraint Map

Given the coordinates of the constraints and the size of the domain, the space is partitioned laterally in the x-direction and longitudinally in the y-direction based on the length and width of each constraint. The z-axis is partitioned for each independent height level of the constraints. The simple constraints shown in Figure 13 result in two partitions along the x and y-axis and a single partition along the z-axis. For constraints modeled as rectangular prisms, this results in a cubed space. Any cube that contains the same space as a constraint is eliminated from the discretized search space.

In the two-dimensional space, a CDT was performed, providing a set of three-sided simplexes. Expanding to three dimensions, four-sided simplexes, or tetrahedrons, are required. To quickly form the simplex discretized space, five tetrahedrons are incorporated into each cube of the discretized search space as shown in Figure 14. By this method, two tetrahedrons occupy the top side of the cube, two tetrahedrons occupy the bottom side of the cube, and a single tetrahedron is placed in the center

of the cube. Each simplex cube is then populated into the discretized space in a checkerboard-like fashion to assure the simplex edges line up on each connecting cube. This results in a three-dimensional discretized space of four-sided simplexes as shown in Figure 14.

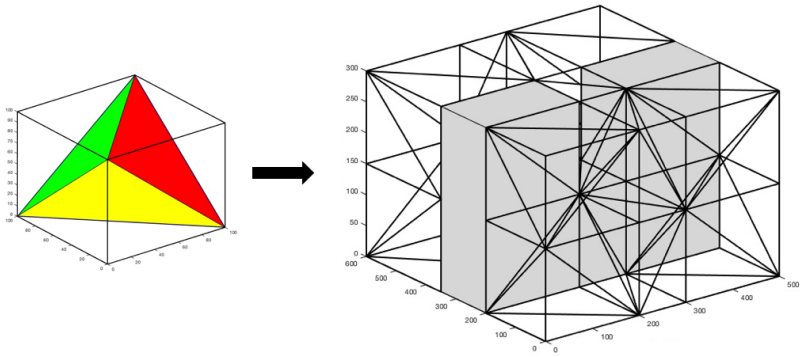

Figure 14. Three Dimensional Discretization

With a discretized space defined, a search method is required to find a CSC containing a feasible path solution from the initial starting point to the terminal point. Similar to the two-dimensional approach, an A* algorithm is implemented based on the equation

$$f(n) = g(n) + h(n). \tag{3.1}$$

Here, $g(n)$ represents the *cost* defined by the Euclidean distance from the mid-point of the current simplex to the mid-point of each connecting simplex. The *cost to go*, $h(n)$, is defined by the Euclidean distance from the midpoint of the connecting simplex to the terminal point of the scenario. The heuristic defined is adequate for this simplified case, however, as the algorithm is developed with a more realistic constraint map, the fidelity of the heuristic implemented should be improved to include flight characteristics of the SUAS. Comparing a Dynamic Programming (DP) search algorithm to an A* approach with the same heuristic, the DP approach results in

the complete search of the space. However, due to the simplicity and symmetry of the constraint map, multiple paths of the same minimal cost exist. To eliminate the subjectivity of choosing one of several paths with the same cost, the A* algorithm is chosen to be implemented, where the first completed CSC identified is accepted.

The A* search algorithm provides a CSC that contains a geometric path solution but may not be feasible due to vehicle dynamics and parameter rate limits. Corrections are made to the resulting CSC to account for two cases where the path is extended unnecessarily due to the constrained field. The first case addresses the situation in which the path re-enters a simplex that is already contained in the path solution, resulting in an infinite loop, and therefore a restriction to enter a simplex already contained in the CSC is enforced. In the second case, the CSC avoids a constraint by entering a set of simplexes congruent to the current CSC. In this case, the congruent simplex corridors have three shared edges, extending the CSC length unnecessarily. These congruent corridors are eliminating, resulting in a minimal set. With these corrections applied, a tetrahedron CSC is defined through the three-dimensional discretized space, free of constraints.

3.5 Optimal Control Problem

The optimal control problem is solved with the direct method using the general purpose optimal control solver GPOPS-II. Inputs are required to define the search domain, accomplished with defined bounds on the state, control, and time vectors. These parameter bounds are often problem specific and subjective in nature, further complicating the problem set-up. Additionally, the optimal control problem must be defined by the objective function, the dynamic constraints, the path constraints, and the event constraints. Each of these inputs have an impact to the quality of the solution returned as well as the time required to converge to a solution. Finally an

initial guess of the states, control, and time is required to seed the NLP. The following sections describe the methodology used to develop the values and functions for each required input to the optimal control solver using a triangular (two-dimensional) or tetrahedron (three-dimensional) formulation.

GPOPS-II Input Parameters.

When defining an optimal control problem in GPOPS-II, the search space is limited to a field defined by the upper and lower bounds placed on the system parameters. These values include the upper and lower limits of the time, states, control, and path constraints. Many times, these values are not intuitive and therefore large values are chosen to assure the search space is not limited (which can cause scaling problems). Additionally, these values are problem specific, requiring parameter adjustments each time the domain of the problem is changed. If these values are not implemented accurately, computation times and convergence issues can become problematic. By discretizing the space with a simplex mesh, these parameters are simplified as the bounds on the domain are limited to a single simplex. Each simplex is solved in GPOPS-II as one phase of the complete solution. In the absence of outside forces, such as wind, the max time limit for each phase becomes the length of the longest simplex edge when solving the minimum time optimal control problem. The state vectors of the SUAS are defined by the barycentric coordinates of the simplex which always range from zero to one for points located inside the simplex and path constraints are eliminated as a result of the triangulation. Finally, the bounds on the control are unchanged as they are vehicle dependent. This results in a standard set of limits that are applied to each problem given a defined CSC through a simplex mesh.

Once the system parameters are defined, the initial guess is the next factor af-

fecting the optimal solution. Due to the gradient-based approach of the software, the better the initial guess that is given to the NLP solver will result in a quicker and more accurate convergence to a solution. The GPOPS-II software requires an initial guess for the time, state, and control vectors. Other optimization software also may require the co-state vector as well. This again can be a difficult task as the co-state vector has no physical relationship to the optimal path. With respect to the GPOPS-II software, providing an initial guess of the time, state and control is not a simple feat. A minimum of two points are required for each of the vectors provided in the initial guess and the accuracy of the solution presented directly impacts the optimal solvers convergence rate.

Coordinate Transformation.

In order to simplify the optimal control solver input parameters, the problem is expressed in the barycentric coordinate system. Given a tetrahedron shown in Figure 15, each vertex is defined in Cartesian coordinates as

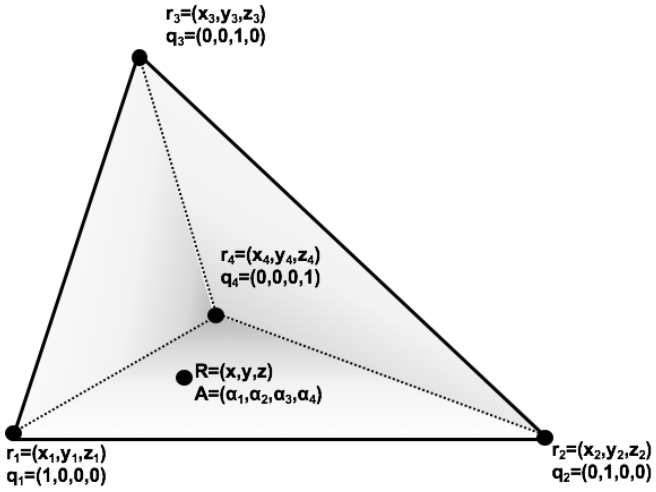

Figure 15. Barycentric Coordinate Frame for a Tetrahedron

$$\mathbf{r_i} = (x_i, y_i, z_i) \;\; \forall i \in [1...n] \tag{3.2}$$

where n is equal to the number of sides of the simplex. The solution is limited to either the two-dimensional plane for which $n = 3$, or the three-dimensional plane for which $n = 4$. The remainder of this section will focus on the three-dimensional approach, where each point within the simplex can be represented as an ordered quartet of real numbers, representing the weighted distribution to each vertex. Each vertex is defined in barycentric coordinates as

$$\mathbf{q_1} = (\mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \tag{3.3}$$

$$\mathbf{q_2} = (\mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{0}) \tag{3.4}$$

$$\mathbf{q_3} = (\mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{0}) \tag{3.5}$$

$$\mathbf{q_4} = (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{1}) \tag{3.6}$$

while any point within the simplex is represented with the corresponding weights to each vertex

$$\mathbf{A} = (\alpha_{\mathbf{1}}, \alpha_{\mathbf{2}}, \alpha_{\mathbf{3}}, \alpha_{\mathbf{4}}). \tag{3.7}$$

Given a set of barycentric weights, the Cartesian coordinates can be represented as

$$\mathbf{R} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \alpha_{\mathbf{i}} \mathbf{r_i}. \tag{3.8}$$

Expanding Equation 3.8, the expression below represents the weights of the barycentric coordinate frame in terms of the Cartesian coordinates,

$$\alpha_{\mathbf{i}} = \mathbf{T}^{-1}(\mathbf{R} - \mathbf{q_N}) \;\; \forall i \in [1...N-1] \tag{3.9}$$

and representing the final weight α_N in terms of the preceding weights, $1 : N-1$,

$$\alpha_N = 1 - \sum_{i=1}^{N-1} \alpha_i \tag{3.10}$$

where N is defined by the number of sides in the simplex and $\mathbf{R} - \mathbf{q_N}$ is a $(N-1) \times 1$ vector summation of the Cartesian coordinates. For the two dimensional problem, $\mathbf{T}$ is a 2×2 matrix representing the vertices of the triangle as

$$\mathbf{T_2} = \begin{pmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{pmatrix}. \tag{3.11}$$

Expanding to three dimensions, $\mathbf{T}$ is a 3×3 matrix defined by

$$\mathbf{T_3} = \begin{pmatrix} x_1 - x_4 & x_2 - x_4 & x_3 - x_4 \\ y_1 - y_4 & y_2 - y_4 & y_3 - y_4 \\ z_1 - z_4 & z_2 - z_4 & z_3 - z_4 \end{pmatrix} \tag{3.12}$$

where x_i, y_i define the vertex locations of each triangle or tetrahedron selected such that the points are not collinear.

GPOPS-II Phased Solution.

GPOPS-II is described as a computational tool for solving multiple-phase optimal control problems using variable-order Gaussian quadrature collocation methods [79]. Each phase is defined with a set of dynamic constraints, path constraints, integral constraints, and parameter constraints. Phases are linked through event constraints, or continuity constraints, that relate information at the start and terminal point of each phase and allow for time, state, and control variables to be continuously transitioned through each phase [81, 25].

For this method, a solution through one simplex can be represented as one phase in GPOPS-II. Formulating the optimal control problem through a simplex set provides the basis for a trajectory solution that traverses through a corridor of simplexes each represented as a single phase, linked together to determine the optimal solution contained inside the defined search space. It is recognized here that the solution found by this method is dependent on the CSC that is presented to the optimal solver. Due to the properties of the A* search algorithm, this CSC may not provide the global optimal solution, rather a local optimal solution may be determined.

The following defines the optimal control problem in terms of a phased approach. Each simplex is represented in barycentric coordinates with appropriate dynamic constraints, path constraints, and parameter constraints. The number of phases required is problem specific and defined after the space has been discretized into a simplex set. The total number of phases in a solution is represented with the variable P.

SUAS Dynamics.

The optimal control problems defined in Chapter IV are developed to show the feasibility, computation efficiency, and accuracy of the CSC approach for both the two-dimension and three-dimensional problem. Chapter V extends the optimal control aircraft model in order to accomplish more difficult mission objectives. In the two-dimensional problem in Chapter IV, the dynamics of the aircraft are represented by the position of the SUAS in the $x(t)$, $y(t)$ directions, the heading angle, $\theta(t)$, and the heading angle rate, $\dot{\theta}(t)$. The control for the aircraft is the change in heading angle rate, $\ddot{\theta}$. Chapter V expands the state vector to include velocity, $v(t)$, and the control vector to include acceleration, $a(t)$. The two-dimensional SUAS dynamics are described below as a relationship between the states and the controls for each phase

(p):

$$\dot{x}^{(p)}(t) = v^{(p)}(t)\cos(\theta^{(p)}(t)) \;\; \forall p \in [1...P] \tag{3.13}$$

$$\dot{y}^{(p)}(t) = v^{(p)}(t)\sin(\theta^{(p)}(t)) \;\; \forall p \in [1...P] \tag{3.14}$$

For scenarios that expand the problem to the third dimension, the dynamics of the aircraft are represented by the position of the SUAS in the $x(t)$, $y(t)$, $z(t)$ directions, the heading angle, $\theta(t)$, and the pitch angle, $\psi(t)$. The control for the aircraft is the change in heading angle rate, $\dot{\theta}(t)$, the change in pitch rate, $\dot{\psi}(t)$, and the velocity, $v(t)$. The three-dimensional SUAS dynamics are described below as a relationship between the states and the controls [82]

$$\dot{x}^{(p)}(t) = v\cos(\psi(t))\cos(\theta(t)) \;\; \forall p \in [1...P] \tag{3.15}$$

$$\dot{y}^{(p)}(t) = v\cos(\psi(t))\sin(\theta(t)) \;\; \forall p \in [1...P] \tag{3.16}$$

$$\dot{z}^{(p)}(t) = v\sin(\theta(t)) \;\; \forall p \in [1...P] \tag{3.17}$$

A summary of the aircraft state and control model for each scenario evaluated is shown in Table 1.

Table 1. Optimal Control Aircraft Models

Optimal Control	**States**								**Control**				
	α_1	α_2	α_3	α_4	θ	$\dot{\theta}$	ψ	v	$\dot{\psi}$	$\dot{\theta}$	$\ddot{\theta}$	v	a
CH. V: 2D	X	X	X		X	X					u		
CH. V: 3D	X	X	X	X	X		X		u_1		u_2	u_3	
CH. VI: ALL (2D)	X	X	X		X	X		X			u_1		u_2

Objective Function.

For the feasibility analysis in both the two and three-dimensional scenarios, the performance measure of each scenario is to minimize the flight time to traverse from

an initial starting point to a specified final location through a series of building constraints and keep-out regions. This is accomplished by advancing through each CSC in a multi-phased approach with the performance measure defined within each simplex by

$$J^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} dt \;\; \forall p \in [1...P] \tag{3.18}$$

and the complete objective formed by summing the flight time within each simplex

$$J = \sum_{i=1}^{P} J^{(p)} \tag{3.19}$$

where t_0 and t_f represent the initial and final time of each phase respectively.

Dynamic Constraints.

With the definition of the barycentric coordinate system and the SUAS dynamics, the dynamic constraints of the optimal control problem can now be defined. The desire is to solve the optimal control problem in a phased approach through a CSC and therefore the dynamics are represented in terms of the barycentric coordinate frame as

$$\alpha_i = f_i(x, y). \tag{3.20}$$

Further illustrated by expanding Equation 3.9 to provide the first two coordinates for a simplex of three sides as follows:

$$\alpha_1 = \frac{(y_2-y_3)(x-x_3)+(x_3-x_2)(y-y_3)}{det(T_2)} \tag{3.21}$$

$$\alpha_2 = \frac{(y_3-y_1)(x-x_3)+(x_1-x_3)(y-y_3)}{det(T_2)} \tag{3.22}$$

The sum of the weights must equal unity, therefore the third coordinate is expressed as

$$\alpha_3 = 1 - \alpha_2 - \alpha_1. \tag{3.23}$$

Evaluating the gradient of f_i defined in Equations 3.21 through 3.23,

$$\dot{\alpha}_i = \nabla f_i \dot{\underline{x}} \tag{3.24}$$

for

$$\underline{x} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.25}$$

yields the dynamic equations with respect to the three-sided simplex. Substituting the aircraft dynamics for the change in the Cartesian x and y positions defined in Equations 3.13 through 3.14 provides the following system of equations

$$\dot{\alpha}_1^{(p)} = \frac{(y_2 - y_3)\dot{x}^{(p)} + (x_3 - x_2)\dot{y}^{(p)}}{det(T_2)} \quad \forall p \in [1...P] \tag{3.26}$$

$$\dot{\alpha}_2^{(p)} = \frac{(y_3 - y_1)\dot{x}^{(p)} + (x_1 - x_3)\dot{y}^{(p)}}{det(T_2)} \quad \forall p \in [1...P] \tag{3.27}$$

$$\dot{\alpha}_3^{(p)} = \frac{(y_1 - y_2)\dot{x}^{(p)} + (x_2 - x_1)\dot{y}^{(p)}}{det(T_2)} \quad \forall p \in [1...P]. \tag{3.28}$$

The state vector for the two-dimensional problem in Chapter IV can now be represented as the barycentric coordinates, heading angle and heading angle rate of the SUAS.

$$X = \left(\alpha_1, \alpha_2, \alpha_3, \theta, \dot{\theta}\right)^T \tag{3.29}$$

Including velocity as a state with control on the acceleration and the change in heading angle rate, the state vector for the two-dimensional problem in Chapter V becomes

$$X = \left(\alpha_1, \alpha_2, \alpha_3, \theta, \dot{\theta}, v\right)^T \tag{3.30}$$

Characterizing a tetrahedron, the barycentric coordinates are expressed as a function of the state variables

$$\alpha_i = f_i(x, y, z), \tag{3.31}$$

further defined as

$$\alpha_1 = \frac{(EI-FH)(x-x_4)-(BI-CH)(y-y_4)+(BF-CE)(z-z_4)}{det(T_3)} \tag{3.32}$$

$$\alpha_2 = \frac{-(DI-FG)(x-x_4)+(AI-CG)(y-y_4)-(AF-CD)(z-z_4)}{det(T_3)} \tag{3.33}$$

$$\alpha_3 = \frac{(DH-EG)(x-x_4)-(AH-BG)(y-y_4)+(AE-BD)(z-z_4)}{det(T_3)}. \tag{3.34}$$

where A through I is defined by the mapping of Equation 3.12

$$\mathbf{T_3} = \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix} = \begin{pmatrix} x_1 - x_4 & x_2 - x_4 & x_3 - x_4 \\ y_1 - y_4 & y_2 - y_4 & y_3 - y_4 \\ z_1 - z_4 & z_2 - z_4 & z_3 - z_4 \end{pmatrix}. \tag{3.35}$$

Again, the sum of the weights must equal unity, therefore the fourth coordinate is expressed as

$$\alpha_4 = 1 - \alpha_3 - \alpha_2 - \alpha_1. \tag{3.36}$$

Evaluating the gradient of f_i defined in Equations 3.32 through 3.34 and 3.36,

$$\dot{\alpha}_i = \nabla f_i \dot{\underline{x}} \tag{3.37}$$

for

$$\underline{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.38}$$

yields the dynamic equations with respect to the tetrahedron resulting in

$$\dot{\alpha}_1^{(p)} = \frac{(EI-FH)\dot{x}^{(p)}-(BI-CH)\dot{y}^{(p)}+(BF-CE)\dot{z}^{(p)}}{det(T_3)} \ \forall p \in [1...P] \tag{3.39}$$

$$\dot{\alpha}_2^{(p)} = \frac{-(DI-FG)\dot{x}^{(p)}+(AI-CG)\dot{y}^{(p)}-(AF-CD)\dot{z}^{(p)}}{det(T_3)} \ \forall p \in [1...P] \tag{3.40}$$

$$\dot{\alpha}_3^{(p)} = \frac{(DH-EG)\dot{x}^{(p)}-(AH-BG)\dot{y}^{(p)}+(AE-BD)\dot{z}^{(p)}}{det(T_3)} \ \forall p \in [1...P] \tag{3.41}$$

$$\dot{\alpha}_4^{(p)} = -\dot{\alpha}_1^{(p)} - \dot{\alpha}_2^{(p)} - \dot{\alpha}_3^{(p)} \ \forall p \in [1...P] \tag{3.42}$$

where the change in the Cartesian x, y, and z positions are defined by the aircraft dynamics in Equations 3.15 through 3.17. The state vector for the three-dimensional problem can now be represented as the barycentric coordinates, heading angle, and pitch angle of the SUAS.

$$X = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \theta, \psi)^T \tag{3.43}$$

Path Constraints.

The equality path constraints for the TOBS problem are defined as the initial and final state of the SUAS. The initial state of the SUAS for phase two of the TOBS mission will be consistent with the final state of the phase one solution. The final

state of the SUAS for phase two will terminate at a target location defined by the user.

The inequality path constraints represent hard constraints in the optimal control problem that cannot be violated. These constraints are formulated as

$$c_{min}^{(p)} \leq c^{p}(y^{(p)}, u^{(p)}, t^{(p)}) \leq c_{max}^{(p)} \ \ \forall p \in [1...P] \tag{3.44}$$

where c is a function of the state, control, and time that influences the vehicle's trajectory. Often, these path constraints represent no-fly zones, terrain, or buildings. Chapter IV will investigate methods to model these constraints while evaluating the accuracy of the flight path and the computation time. The primary method discretizes the domain with a simplex mesh such that constraints can be represented as polygons and isolated from the desired CSC upon which the path solution will be found, effectively eliminating the constraint from the search domain of the NLP.

Integral Constraints.

Integral constraints are minimized in the cost function and represent soft constraints in the optimal control problem. They are formulated as inequalities defined by

$$g_{min}^{(p)} \leq \int_{t_0^{(p)}}^{t_f^{(p)}} g^{(p)}(y^{(p)}, u^{(p)}, t^{(p)}) \leq g_{max}^{(p)} \ \ \forall p \in [1...P]. \tag{3.45}$$

These constraints can be used to minimize the incursion to keep-out regions, fuel consumption, or the control authority of the SUAS. Chapter V evaluates a scenario in which keep-out regions are placed along the CSC, requiring the SUAS to traverse into the region while minimizing the incursions to the space.

Event Constraints.

Event constraints in GPOPS-II, also referred to in literature as phase or continuity constraints, are implemented to maintain a continuous transition of the state variables between each phase and are expressed as

$$X_0^{(p)} - X_0^{(p-1)} = 0 \ \ \forall p \in [2...P] \tag{3.46}$$

where X represents the state vector. Examining the two-dimensional case, this condition requires that each vertex of a simplex be defined as the q_1, q_2, or q_3 vertex. As the path trajectory traverses across an edge, one of the weights (states 1-3) will be zero as the weight associated with the opposing vertex has no contribution to the location of the point. As the new phase begins, it is imperative that the states of the next simplex match the states of the previous simplex. In other words, the opposite vertex of the new simplex must accept the zero value and the associated weights for the other vertices must match appropriately in the state vector. This is illustrated in Figure 16.

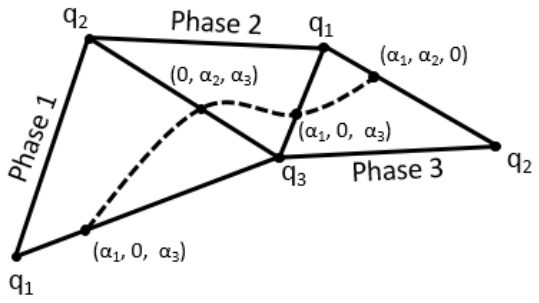

Figure 16. Event Constraints through Notional Simplex Corridor

Extending this methodology to the third dimension, the path will transition from one simplex to the next through a shared face. The three shared vertices on this face can take a barycentric weight value between zero and one. However, the fourth

vertex on each simplex must accept a zero value as it provides no weight to the path position when located at a simplex transition boundary. Implementing these bounds within the optimal control solver and defining the zero state for each phase increases the computation speed by providing a required direction for the search.

Bounds.

The search space of the optimal solver is limited by the bounds applied to the state variables, the control, and the mission time. Often, these bounds can be difficult to determine. If they are set too large, convergence times can become excessive. If they are set too small, there is a greater probability the solution will converge to a local minimum instead of continuing to search for the global minimum. Additionally, if the bounds are set too restrictive, the optimal solution may no longer be in the search space and a feasible solution may not be found.

By defining the problem with a simplex set in the barycentric coordinate system and solving each simplex as a separate phase, the bounds become simplified and strictly defined for each problem. The weights of the barycentric coordinates are defined from 0 to 1, therefore the bounds on the position states become

$$0 \leq \alpha_i \leq 1 \;\; i = 1, 2, ...N, \tag{3.47}$$

where N defines the number of edges in the simplex. The time vector is bounded with an upper and lower limit in each phase. For the simplistic minimum time path solution, the furthest distance the SUAS can travel through any simplex is equal to the length of the longest edge, defined as $edge_{max}$. The max time bound is then determined as the time required to fly along the longest edge at the minimum speed,

$$0 \leq t \leq \rho \frac{edge_{max}}{v_{min}}. \tag{3.48}$$

Here, ρ represents a scaling factor to allow for extended time in the presence of winds or other disturbances, as accomplished in Chapter V.

The bounds on the remaining state and control parameters are specific to vehicle characteristics and are set such that a tractable scenario can be accomplished.

Algorithm Development.

The flow chart in Figure 17 describes the series of algorithms required to achieve a solution to the optimal control problem using simplex discretization. This flow chart illustrates the hybrid method, combining the fast geometric path solutions attained by Triplanner with direct orthogonal collocation methods for acquiring optimal path solutions.

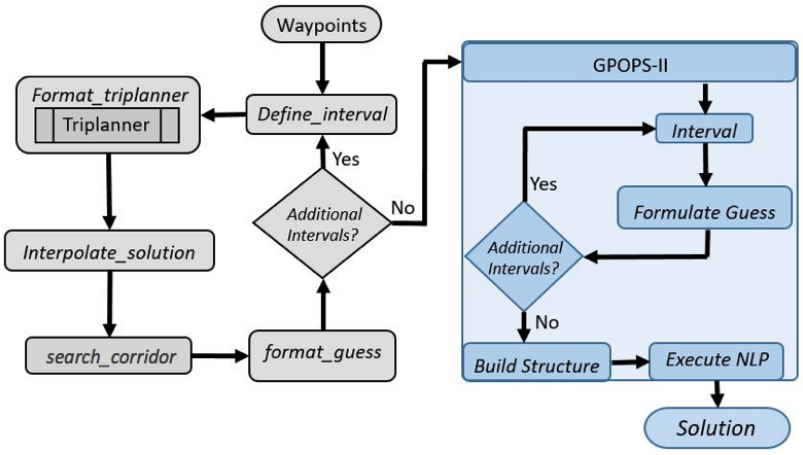

Figure 17. Algorithm Flow Chart

The *Waypoints* algorithm shown in the flow chart is initiated with the function *Define_interval*. Here, the number of waypoints in the path solution is defined along with a matrix consisting of each waypoint location. The first and last waypoints of the matrix consists of the starting location of the SUAS, thus requiring the vehicle to return to its original location. An interval is defined as the path between each waypoint location resulting in $N + 1$ intervals for N waypoints. For scenarios with

only one terminal point, the vehicle will only fly one interval, from the start position to the terminal waypoint.

The *Format_triplanner* algorithm selects the start and end point for the current interval, consisting of a set of phases, and initiates the Triplanner algorithm. The output of Triplanner provides text files consisting of the constrained and unconstrained edges in the simplex mesh forming the CDT as well as the Dubins path solution. Figure 18 shows the resulting geometric path consisting of constant radius turns connected by straight line segments. The blue asterisks illustrate the results of the Triplanner solution. Constrained edges are illustrated with solid red lines, while unconstrained edges are solid gray lines. The path solution starts at the lower left green asterisk and ends at the upper right red asterisk.

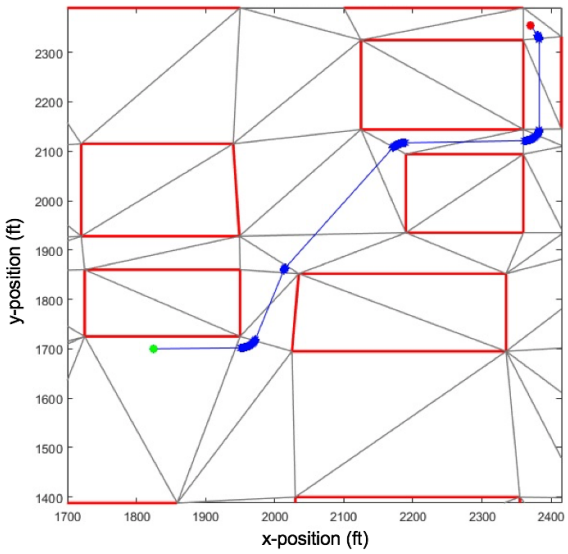

Figure 18. Triplanner Solution

With the resulting path from Triplanner, points are interpolated along the straight line sections of the path solution. This is accomplished based on a predefined spacing in the *interpolate_solution* function. This provides a foundation for dividing the path solution into each represented simplex and constructing a properly formatted initial

guess for the optimal control solver.

The function *search_corridor* performs three tasks. First, the interpolated path is transformed to barycentric coordinates and the path is segmented into each simplex. Second, a connectivity matrix is formed defining the order of simplexes that contain the path solution. Here, the vertex points of each simplex are arranged such that the common edge between consecutive simplexes can be identified. Finally, the connectivity matrix is augmented with three columns defining the shared simplex edge along the CSC, the determinant of matrix T_2 found in Equation 3.11, and the time required to traverse the longest edge of each simplex. Figure 19 shows the interpolated path illustrated with blue asterisks and the CSC shown with black solid lines.

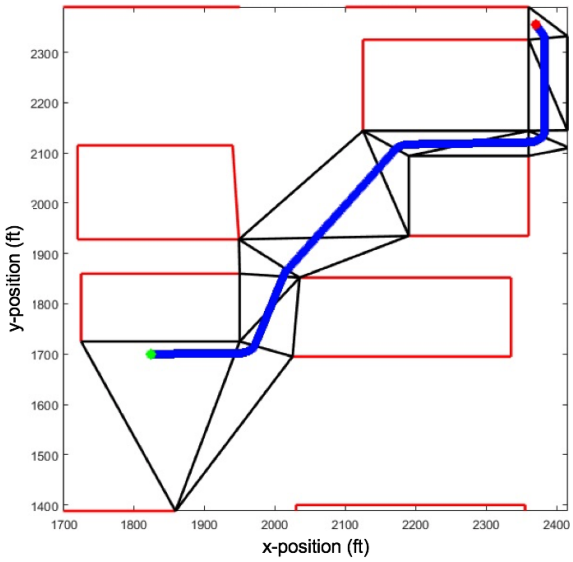

Figure 19. Triplanner Interpolated Path Solution

The final function block before entering the optimal solver, *format_guess*, builds the required vectors for an appropriate guess for the optimal control problem. The heading is formulated using a three-point finite differencing scheme beginning with the position vector of the interpolated path solution. Since the Triplanner results in a Dubins path, the heading rate and change in heading rate are defined by the

SUAS rate limits based on the heading angle. For scenarios where velocity is included as a state, the velocity vector is formed with a maximum speed during straight line portions of the path and minimum speed during the constant radius turns. This allows for Triplanner to investigate a solution in a larger search domain as the slower SUAS speeds result in a tighter turn radius allowing the vehicle to traverse through highly constrained regions. The rate limited acceleration vector is then formed based on the change in the velocity curve. These vectors are shown in Figure 20.

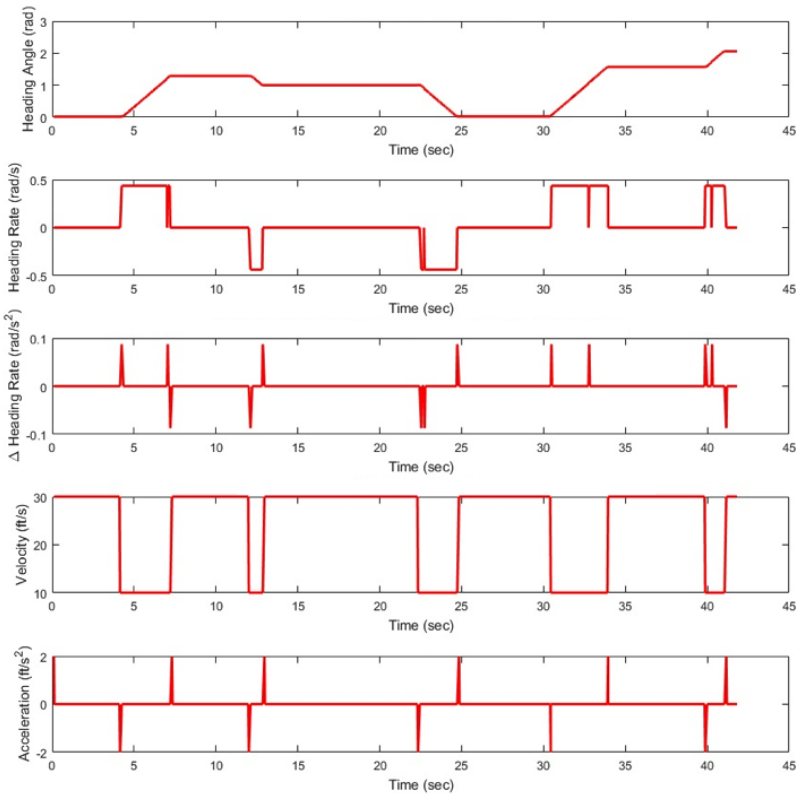

Figure 20. Initial State and Control Vectors

Each of these vectors are described in a three tier structure representing the defined interval and phase of the solution. This process is repeated for each interval with the structure being appended to the end of the previous interval. Once all target intervals have been exhausted, the connectivity and guess structures are output to the optimal control software.

3.6 Summary

This section provided an overview of the proposed methodology used for this research. The SUAS dynamics were defined for both the two and three-dimensional problems and the general optimal control problem was developed. A methodology was developed to eliminate subjective bounds on problem specific parameters and path constraints were removed from the search domain by formulating the problem within the construct of a simplex or tetrahedron discretization and transforming the vehicle position to barycentric coordinates. The Triplanner toolkit algorithms were described and will be used to seed the NLP solver when solving for two-dimensional optimal path solutions while the discretization method for the three-dimensional case was explored.

IV. Constrained Optimization Approach

4.1 Overview

This chapter evaluates the methodologies presented in Chapters II and III. First, three different types of constraint functions are modeled in the optimal control solver, to include simple circular and elliptical shapes, ellipsoids, and polygon shapes. These results are compared with simplex discretization methods upon which polygonal constraints are eliminated from the search field. Second, the simplex method is further developed in a two-dimensional scenario to determine a feasible flight path of a single SUAS traversing through a constrained environment in downtown Chicago. Finally, the concepts of the two-dimensional problem are extended to three dimensions, where a SUAS is required to fly around or over three rectangular prism constraints. Each method presented is evaluated for accuracy of the solution and the computational speed of the optimal solver, GPOPS-II.

4.2 Constraint Analysis Simulation Overview

To evaluate the presented constraint models, a two-dimensional optimal control problem was developed. For each scenario in the constraint analysis, the initial and final aircraft state values are defined as

$$(x_0, y_0, \theta_0) = (1, 0.5, free) \tag{4.1}$$

$$(x_f, y_f, \theta_f) = (9, 9.5, free). \tag{4.2}$$

Three constraints, which represent areas the SUAS can not traverse, such as buildings, walls, terrain, or restricted air space are included. These constraints are modeled as polygons inside the domain space.

Six simulations were evaluated in total. The first three were performed without triangulation and modeled the constraints with traditional methods of circular and elliptical shapes, superquadrics, and polygon functions. The fourth method evaluates a technique that discretized the space through a CDT and optimizes a path through a CSC. The initial guess for this method is the path connecting consecutive midpoints of each unconstrained edge in the CSC. The fifth method examines the path solution from the Triplanner toolkit which provides a solution based on a heuristic search algorithm. The final method combines Triplanner and GPOPS-II, using the Triplanner solution as an initial guess for the optimal control solution in GPOPS-II. Each model is compared with computation time, objective time, mesh tolerance, and initial guess requirements.

The first simulation models the constraints with simple circular and elliptical shapes where solutions can result in quick convergence to the optimal trajectory. However, large errors will exist when representing polygons with these simple shapes. Further, when solving with collocation methods such as GPOPS-II, collocation points could jump through a constraint dependent on the spacing of the collocation points. Figure 21A below illustrates a failed solution based on a poor guess. Although the NLP solver reported an optimal solution, it is not a valid flight path around the constrained environment. As the guess is improved in Figure 21B, an optimal solution is found, however the path continues to violate the constraint as shown in Figure 21C.

In the second simulation, superquadrics are used to better represent polygonal shapes such as squares or rectangles by increasing the power, m, in Equation 2.19. Figure 22A below shows the same constraint field as the previous simulation, however now the polynomial order of the constraint function has been raised to 100. A feasible solution could only be found when the mesh tolerance was increased to 10^{-5} and a perfect initial guess was given to the NLP solver.

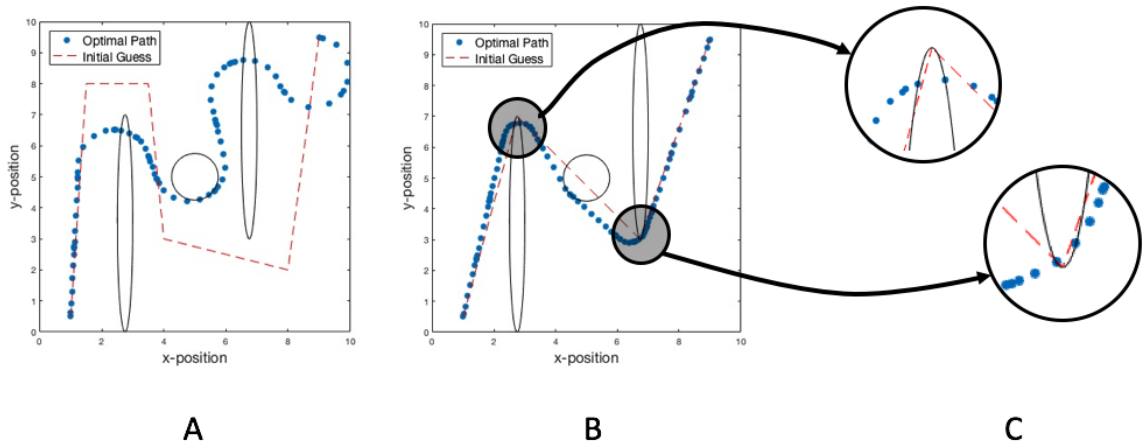

Figure 21. Simple Shape Constraint Functions; Illustrating Dependency of Quality Initial Guess

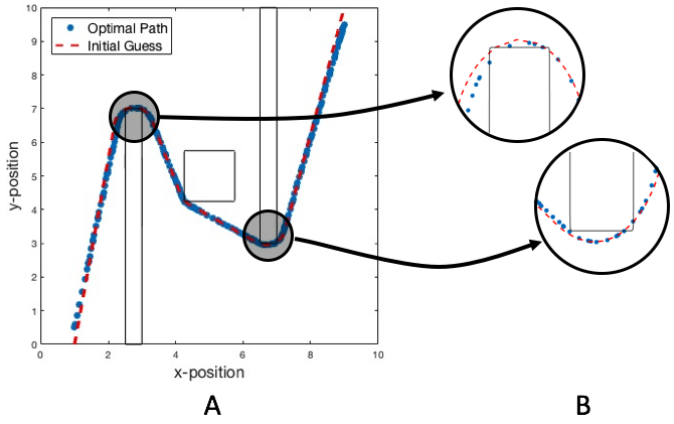

Figure 22. Superquadric Constraint Functions; Illustrating Converged Solution for Small Semi-major Axis with Quality Initial Guess

These constraint functions are more representative of polygonal shapes, however the flight path still cuts the corner of the constraint as the collocation points round the corner of each constrained edge as shown in Figure 22B. Further, the solution still requires a perfect guess and computation time remains between 10 and 20 seconds.

The third simulation utilizes boolean function which allows for the modeling of any polygonal shape. Solutions presented in GPOPS-II are feasible, but the required computation time exceeded the limits for onboard operations. Additionally, as more

constraints are added to the optimal control problem, the computational requirements become unwieldy.

Ultimately, the convergence of a solution is highly dependent on the user settings of GPOPS-II. The mesh tolerance can be increased to allow for tighter discretization of the space and to include more collocation points, however, this increases computation time and may result in convergence failure if the mesh is too tight for the given problem. Additionally, the initial guess plays a significant role in the solution search space. A poor initial guess may result in a local minimum solution rather than the global solution. Further, the better the guess the solver receives, the quicker the convergence to a solution. Finally, the shape of the constraints have an effect on the feasibility of the solution as well. Collocation points may cut the corner of the edge of a constraint or may even skip over the constraint if the collocation points are not spaced properly. Each of these factors are problem specific and require different dependencies on the GPOPS-II user settings. The next three simulations focus on eliminating these constraints from the problem formulation while minimizing the input requirements to achieve a feasible solution within computation times sufficient for onboard operations.

The fourth simulation triangulates the space using the CDT function, *delaunay*, in MATLAB®. A CSC is selected with dynamic programming and an initial guess is determined by connecting the midpoint of each unconstrained edge of the CSC. The solution is shown in Figure 23.

In the left image, it can be seen that the optimal flight path determined through the CSC does not violate any hard constraint. The right image illustrates the heading rate control over the flight time of the simulation. The control can be further refined to fully exhibit Pontryagin's Minimum Principle by increasing the collocation points and the mesh tolerance used in the simulation.

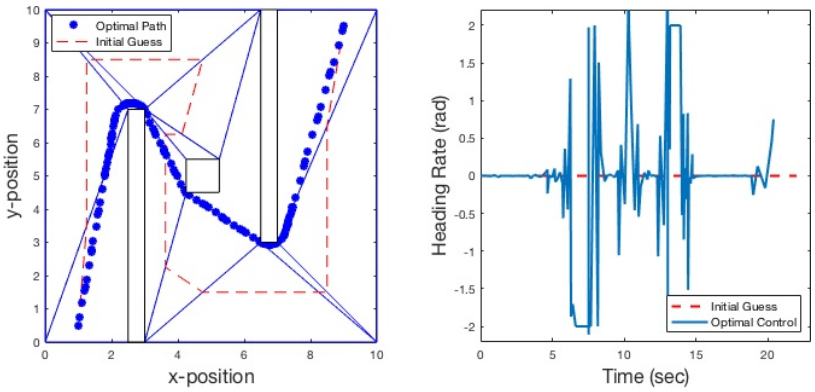

Figure 23. CDT Path Solution & Control; Mid-point Guess

The fifth solution method utilizes the Triplanner toolkit developed by Kallmann. The algorithm is built in C++ and accessed through a Python script run on a Linux operating system.

The data required to initialize the Triplanner algorithm consists of the points of each closed polygonal constraint, the initial and final starting points of the path, and the offset distance required from each constrained edge. In order to achieve a feasible path for a SUAS, the offset distance was set to the minimum turning radius of the aircraft. The Triplanner algorithm returns a Dubins path solution with computation time on the order of milli-seconds. The discretized solution that is returned is shown in Figure 24.

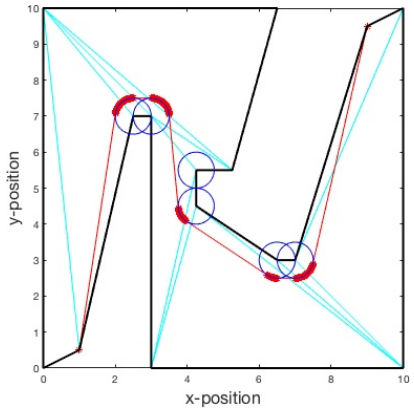

Figure 24. Constraint Analysis Triplanner Toolkit Solution

Here the Dubins path solution can be seen as a series of straight lines and constant radius turns. Given the position coordinates, and a solution satisfying Pontryagin's Minimum Principle, the heading and heading rate can be determined for the two dimensional case with simple SUAS dynamics with a three-point finite differencing scheme.

The sixth simulation uses the Triplanner solution as the initial guess to the NLP solver. The Triplanner toolkit returns the discretized path solution, the CDT, and the CSC. Implementing this data into the triangulation method with GPOPS-II a hybrid solution is formed for which the optimal solution is obtained and is presented through the given CSC shown in Figure 25.

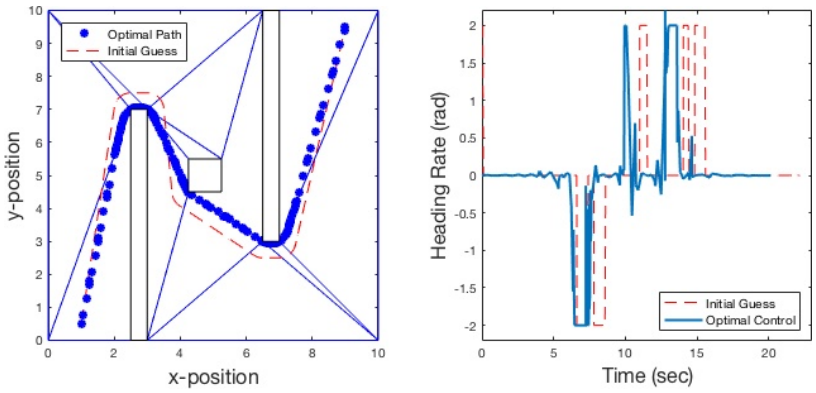

Figure 25. Hybrid Path Solution & Control

The solution is optimal given the requirement that the path must traverse through each triangle of the CSC in the same order as the initial guess. Again there can be seen some variance in the control due to the mesh tolerance setting of 10^{-2}. These settings allowed for a comparable number of collocation points to the other simulations.

4.3 Constraint Analysis Simulation Results

For each simulation model, the mesh tolerance, computation time, objective time, collocation points, and the quality of the initial guess were recorded. The NLP solver

in each of these simulation was "SNOPT". The results for each method are displayed in Table 2.

Table 2. Constraint Analysis Simulation Results

Constraint	Mesh	Comp Time (s)	Path Length (ft)	Discretized Pts	Guess
Simple Shapes	10^{-4}	4.40	19.39	104	Good
Superellipse	10^{-5}	21.34	20.12	211	Perfect
Polygons	10^{-5}	9.96	20.68	159	Perfect
Triangulation	10^{-2}	17.37	20.14	221	Poor
Triplanner	N/A	$2.7x10^{-3}$	22.6	377	N/A
Hybrid	10^{-2}	2.91	20.12	221	Tripath

The Triplanner toolkit solution solved in 2.7 milli-seconds but also had the longest path length, does not include vehicle dynamics, and does not directly return the states and control of the vehicle, preventing the inclusion of more complex models. The hybrid solution solved with the fastest convergence time in GPOPS-II at 2.91 seconds with a performance index equivalent to the other optimal control methods.

Although the path length is compared to the first three methods, the problems that are solved differ by the way the constraints are formulated, resulting in a different search space. Therefore, the path lengths should not be compared directly, but rather used to assure each method is in close proximity to each other. The method solved with simple shape constraints returned the smallest value for the path length, but did not adequately model a polygon constraint. The remaining methods, modeling constraints as polygons, each returned consistent values for the path length.

When designing constraints composed of rounded edges, the triangulation method will require a polygon fit either to the interior or exterior of the rounded edge, dependent on the type of constraint being modeled. Soft constraints can be modeled to the interior of the rounded edge, allowing for the path to cut through small portions of the constraint. Hard constraints should be modeled on the exterior of the rounded edge where the model error will result in a longer path solution while eliminating the constraint from the solution space entirely. In either case, modeling errors can be

minimized by increasing the polygonal vertices used to model the constraint, however each additional vertex requires additional phases in the optimal control problem.

4.4 Two-Dimensional Optimal Control

By discretizing the search space in a simplex mesh, a phased approach is taken in the optimal control problem to extend the methodology to a more realistic and complex scenario. The dynamics are formulated in Cartesian coordinates, representing the position of the SUAS in the $x(t)$, $y(t)$ directions and the heading angle θ defined as

$$\dot{x}^{(p)}(t) = v\cos(\theta^{(p)}(t)) \;\; \forall p \in [1...P] \tag{4.3}$$

$$\dot{y}^{(p)}(t) = v\sin(\theta^{(p)}(t)) \;\; \forall p \in [1...P] \tag{4.4}$$

where v represents the aircraft velocity which is held constant in this scenario at $30ft/s$.

Optimal Control Problem.

Each triangle is solved as a single phase in GPOPS-II and each phase is connected through event constraints. The dynamics constrain the path through each triangle in barycentric coordinates. The optimal control problem formulation has been consolidated as follows.

Minimize the cost functional

$$J = \sum_{p=1}^{P} J^{(p)} \tag{4.5}$$

where

$$J^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} dt \;\; \forall p \in [1...P] \tag{4.6}$$

subject to the dynamic constraints

$$\dot{\alpha}_1^{(p)}(t) = \frac{(y_2-y_3)v\cos(\theta^{(p)}(t))+(x_3-x_2)v\sin(\theta^{(p)}(t))}{det(T)} \;\; \forall p \in [1...P] \tag{4.7}$$

$$\dot{\alpha}_1^{(p)}(t) = \frac{(y_3-y_1)v\cos(\theta^{(p)}(t))+(x_1-x_3)v\sin(\theta^{(p)}(t))}{det(T)} \;\; \forall p \in [1...P] \tag{4.8}$$

$$\dot{\alpha}_3^{(p)}(t) = -\dot{\alpha}_1^{(p)}(t) - \dot{\alpha}_2^{(p)}(t) \;\; \forall p \in [1...P] \tag{4.9}$$

with the control on the change in heading rate

$$u^{(p)}(t) = \ddot{\theta}(t) \tag{4.10}$$

and the state vector defined with a five state model in barycentric coordinates as

$$\mathbf{X} = (\alpha_1, \alpha_2, \alpha_3, \theta, \dot{\theta}) \tag{4.11}$$

with boundary conditions given as the initial and final constraints,

$$\mathbf{X}^{(1)}(t_0^{(1)}) = ((\alpha_1)_0, (\alpha_2)_0, (\alpha_3)_0) \tag{4.12}$$

$$\mathbf{X}^{(P)}(t_f^{(P)}) = ((\alpha_1)_f, (\alpha_2)_f, (\alpha_3)_f,). \tag{4.13}$$

Inequality path constraints representing bounds on the state, control and time are

defined as

$$0 \leq \alpha_1^{(p)}, \alpha_2^{(p)}, \alpha_3^{(p)} \leq 1 \tag{4.14}$$

$$|\theta^{(p)}| \leq 180 \ \ deg \tag{4.15}$$

$$|\dot{\theta}^{(p)}| \leq 25 \ \ deg/s \tag{4.16}$$

$$|u^{(p)}| \leq 2 \ \ deg/s^2 \tag{4.17}$$

$$0 \leq t^{(p)} \leq \frac{edge_{max}^{(p)}}{v}, \tag{4.18}$$

where $edge_{max}$ represents the longest edge in the defined simplex. Equality path constraints are implemented in a scenario to allow the SUAS to maintain a safe distance from each constrained edge of the CSC,

$$\left(r_{1x}^{(p)} - x^{(p)}(t)\right)^2 + \left(r_{1y}^{(p)} - y^{(p)}(t)\right)^2 - (\delta)^2 = 0 \tag{4.19}$$

$$\left(r_{2x}^{(p)} - x^{(p)}(t)\right)^2 + \left(r_{2y}^{(p)} - y^{(p)}(t)\right)^2 - (\delta)^2 = 0 \tag{4.20}$$

$$\left(r_{3x}^{(p)} - x^{(p)}(t)\right)^2 + \left(r_{3y}^{(p)} - y^{(p)}(t)\right)^2 - (\delta)^2 = 0 \tag{4.21}$$

where r_i represents the x and y coordinate of simplex vertices and δ defines the minimum safety buffer between the SUAS and the building constraints defined by the distance between each vertex point and the flight path. An alternative solution, evaluated in a separate scenario, incorporates this safety buffer into the polygon constraint itself to illustrate the benefits of eliminating path constraint functions from the optimal control problem. Finally, event constraints are included to maintain a continuous transition of the state variables between each phase,

$$X_o^{(p)} - X_f^{(p-1)} = 0 \ \ \forall p \in [2...P]. \tag{4.22}$$

4.5 Two-Dimensional Scenario

This two-dimensional scenario illustrates a solution for a single SUAS flying through downtown Chicago with an altitude constraint of $600ft$ Above Ground Level (AGL). The scenario is chosen to represent a challenging urban environment to verify the functionality of the solution method. The search space is defined by a small region of downtown Chicago measuring $5600ft$ by $2800ft$. All structures within this region that exceed $550ft$ AGL are modeled with a simplistic four-sided polygon. For this scenario, the initial starting point for the SUAS is a parking garage located on the south-west side of the city, while the final location is a monument located on the north-east side of the city. The initial and final aircraft constraints are defined as

$$(x_0, y_0, \theta_0) = (200, 200, free) \tag{4.23}$$

$$(x_f, y_f, \theta_f) = (4700, 2650, free). \tag{4.24}$$

Within the defined search space, there are 37 buildings that exceed $550ft$. Figure 26A shows the constraint map with the initial and final flight coordinates illustrated with a green and red asterisk respectfully. Performing a CDT on the space, Figure 26B illustrates the discretized mesh defined by the constrained environment. Each building taller than $550ft$ is represented on the map as a red, four sided polygon.

Utilizing the 2010 version of the Triplanner toolkit developed by Kallmann and his team, the CSC is defined and a feasible flight path solution is determined. The data required to initialize the Triplanner algorithm consists of the points of each closed polygonal constraint, the initial and final starting points of the path, and the offset distance required from each constrained edge. In order to achieve a feasible path for a SUAS, the offset distance was set to the minimum turning radius of the aircraft determined through the relationship between the aircraft's velocity and turn rate as

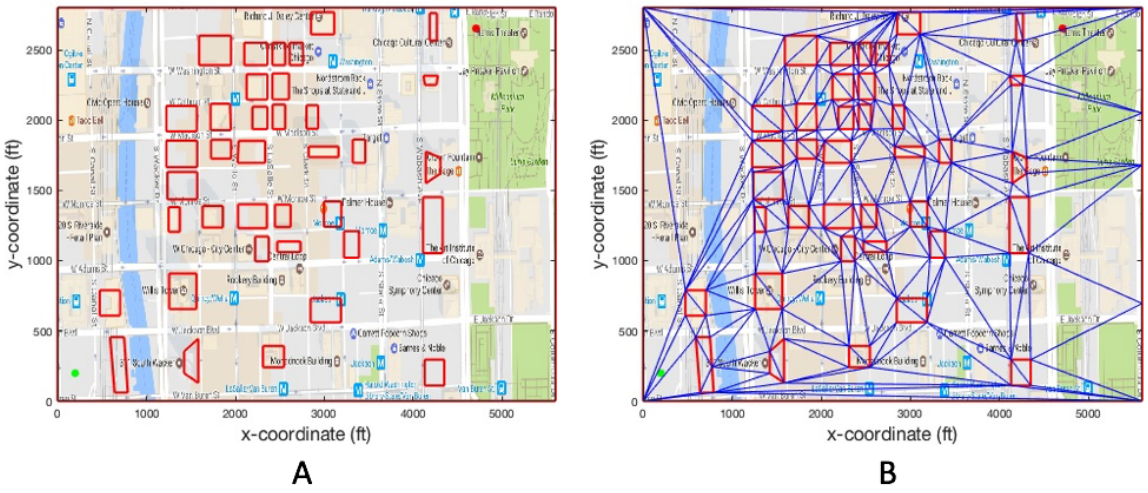

Figure 26. Downtown Chicago; 550ft AGL Constraint Map (A), Discretized Simplex Mesh (B). [Map data @2017 Google]

follows,

$$\tilde{r} = \frac{v}{\dot{\theta}_{min}} \tag{4.25}$$

for $\tilde{r}$ is the minimum turn radius, v is the velocity, and $\dot{\theta}$ is the turn rate. The Triplanner algorithm returns a Dubins path solution with computation time on the order of milli-seconds. The CSC and the path solution returned from the Triplanner toolkit is shown in Figure 27.

The Triplanner solution is solved in 5.84 milliseconds, with a path length of $5490.9ft$, however, the aircraft dynamics and control are not incorporated into the Triplanner toolkit as the Dubins path solution is found through a discretized A* search algorithm based on the geometric properties of the problem formulation. The output of the Triplanner algorithm provides the triangulated mesh that includes all the constrained and unconstrained edges as well as the discretized path solution. This data is interpolated to provide a solution comprised of data points at $1ft$ spacing. This minimal spacing is required to assure that data points exist in each simplex of

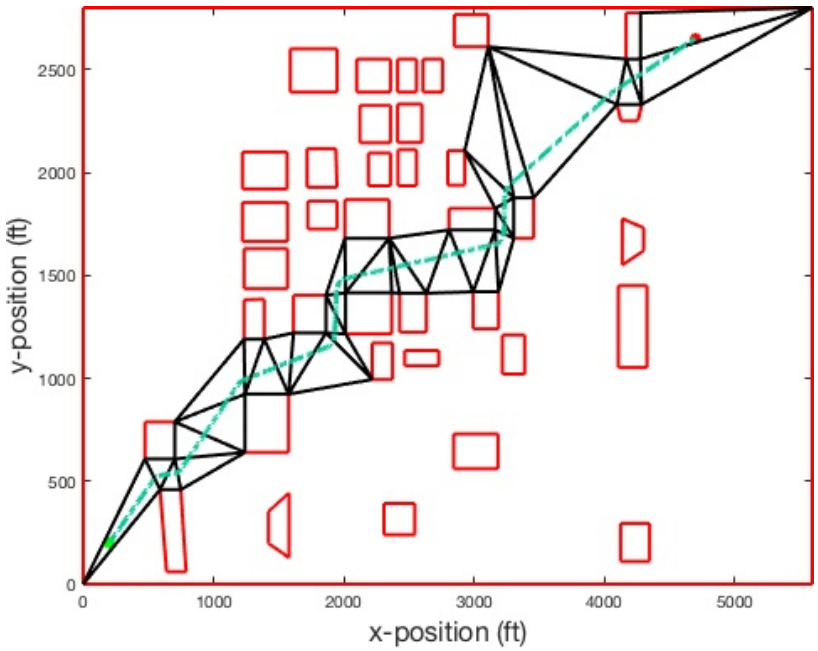

Figure 27. Two-Dimensional Triplanner Toolkit Path Solution

the CSC.

Two simulations were executed to illustrate the necessity of providing a quality guess to the optimal control software. For each simulation, the optimal control problem is solved in GPOPS-II under the same set-up parameters with only the initial guess for the state vector, control, and time differing. The key GPOPS-II parameters are listed below in Table 3.

Table 3. GPOPS-II User Settings, 2D Scenarios

GPOPS-II User Settings	
Mesh Method	hp-PattersonRao
Mesh Tolerance	10^{-2}
NLP Solver	SNOPT
Derivative Supplier	AdiGator
Method	RPM-differential
NLP Tolerance	10^{-3}
Min Collocation Points	4
Max Collocation Points	10
Mesh Fraction	$\frac{1}{2}$*ones(1,2)
Mesh Collocation Points	4*ones(1,4)

The minimum flight safety buffer, δ, preventing an aircraft from flying too close to

a building is set to $15ft$ for each simulation. The path results for the first simulation are shown in Figure 28.

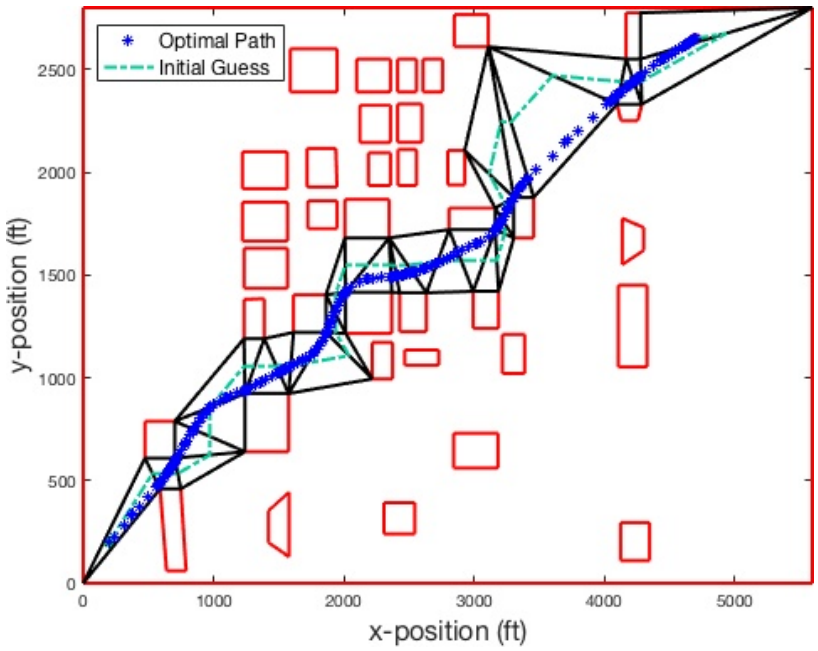

Figure 28. Two-Dimensional Optimal Path Solution; Mid-Point Guess

The initial guess is illustrated with the dotted green line while the discretized path solution is shown with the dark blue asterisks. An optimal solution through the CSC was found with a computation time of 4.57 seconds. The heading angle and the heading angle rate are the fourth and fifth states and are displayed in the top two plots of Figure 29. The control is shown in the lower plot of Figure 29 with an initial guess equal to the zero vector.

Evaluating these plots, the mid-point solution requires a heading change at the start of each simplex. This requires excessive vehicle control inputs when compared to the optimal solution. The second simulation provides an initial guess vector for the state, control, and time as determined for each triangle through the Triplanner toolkit solution and implemented in GPOPS-II as individual phases. The results are shown in Figure 30.

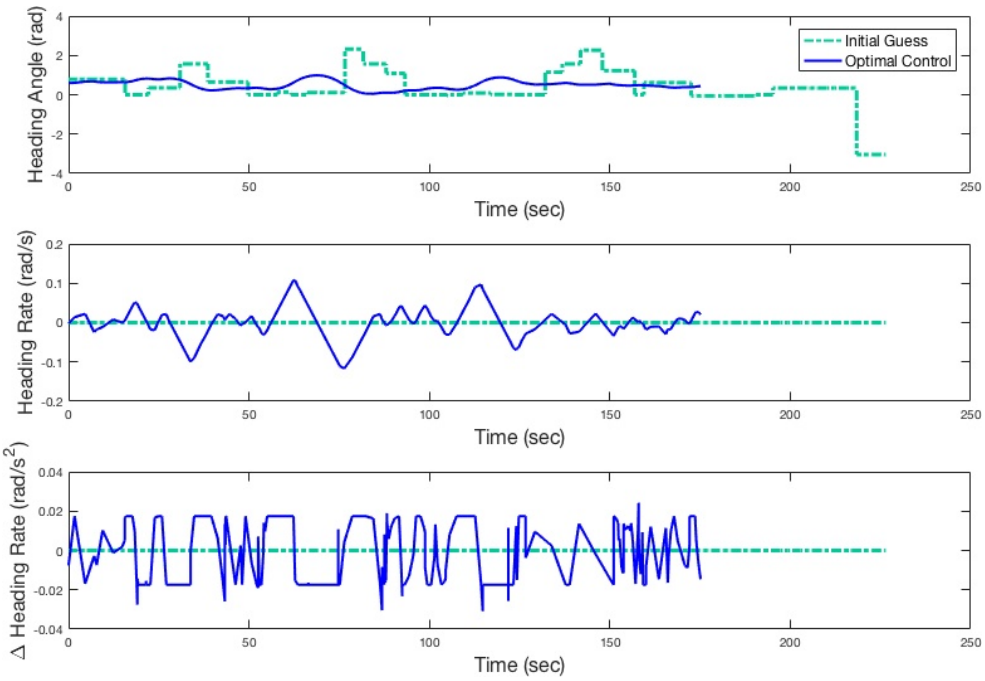

Figure 29. Two-Dimensional Optimal Path Solution & Control; Mid-Point Guess

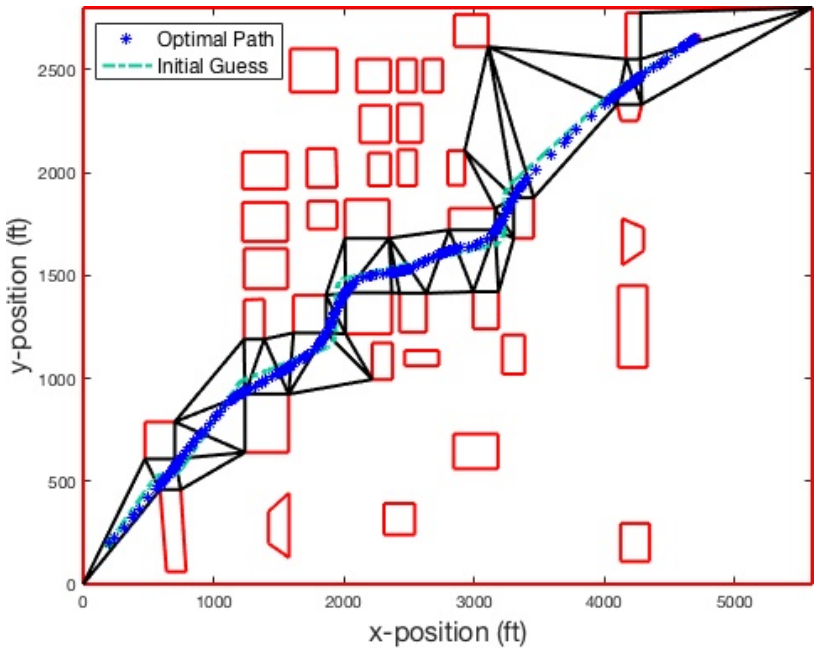

Figure 30. Two-Dimensional Hybrid Path Solution

Again, the initial guess is illustrated with the dotted green line while the discretized path solution is shown with dark blue asterisks. An optimal solution through

the CSC was found, however, given a quality initial guess to the path solution, the computation time for the optimal control software decreased to 3.58 seconds. The heading angle, heading angle rate, and control are shown in Figure 31 with the initial guess resulting from the Triplanner toolkit solution shown with the green dotted line and the dark blue solid line describing the optimal heading angle and control.

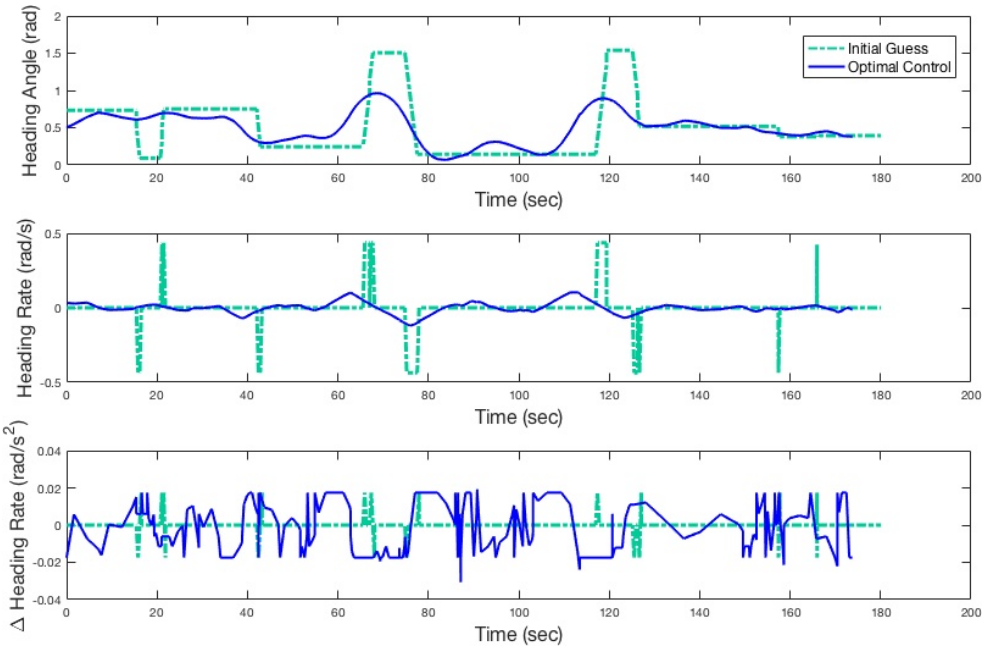

Figure 31. Two-Dimensional Hybrid Path Solution & Control

Here the angle requirements between the Triplanner solution and the optimal control solution have been minimized and when compared to the mid-point solution, the Triplanner solution requires more control than the optimal solution but significantly less than the mid-point solution.

4.6 Two-Dimensional Conculsions

The two simulations shown above were each solved with an NLP tolerance of 10^{-3}. Through simulation, it was determined that decreasing the NLP tolerance to a lower threshold significantly increased the computational time while improving the

objective value by just a few tenths of a second. To further decrease the computational time, the minimum flight safety buffer, defined in Equations 4.19 - 4.21, can be removed from the optimal control equality constraints. This safety buffer can instead be incorporated inside the polygon of the original building constraints resulting in equivalent objective costs while reducing the computation time to 2.12 seconds. Table 4 shows the Triplanner only results for computational time and path length for each set of constraints, one with (NLP Equality Constraints) and one without (In-Polygon Model) the safety buffer.

Table 4. Triplanner Results

Building Safety Buffer	**Path Length (ft)**	**Comp Time (ms)**
NLP Equality Constraints	5398.9	4.94
In-Polygon Model	5490.9	5.84

Table 5 shows the computation time of GPOPS-II, objective time, and path length for the optimal control solution. These results illustrate the significant differences in computation time that can be attained by incorporating the Triplanner algorithm as the initial guess to the NLP and maintaining the building safety buffer inside the polygon building constraint model.

Table 5. Two-Dimensional Simulation Results

Building Safety Buffer	**Initial NLP Guess**	**Comp Time (s)**	**Obj Time (s)**	**Path Length (ft)**
NLP Equality Constraints	Mid-Point	4.57	173.26	5210.8
NLP Equality Constraints	Triplanner	3.58	173.63	5212.1
In-Polygon Model	Mid-Point	3.84	174.32	5256.6
In-Polygon Model	Triplanner	2.12	173.69	5240.9

Without a CSC, an initial guess to the NLP for this simulation would be difficult to generate and path constraints would be challenging to model, resulting in an optimal solution that would likely return a non-desired local minimum with excessive

computation times. Here it is shown the optimal control problem can be solved in 2.12 seconds with an objective cost of 173.69. This solution requires an initial guess acquired from the Triplanner toolkit solution for the safety buffer modeled inside the polygon constraint. The Triplanner solution alone solved in the fastest time at 5.84 milliseconds, with a path length of $5490.9ft$. Although this solution is well within the computational limits for onboard processing, it requires an additional 280 feet of path length to accomplish the mission and the state and control parameters are not explicitly returned.

4.7 Three-Dimensional Optimal Control Problem

A few challenges must be overcome to transition the two-dimensional concepts to the third dimension. First, the Triplanner toolkit is a fast geometric path planner built only for two-dimensional space. In order to extend the methodology to allow for pitch control of the SUAS, the three-dimensional space must be discretized into tetrahedrons. Currently, there is not software available to perform this discretization quickly and efficiently and therefore a simple constraint model is used in this simulation such that a feasibility analysis can be conducted. Once the simplex domain is achieved, a CSC can be determined through an A* algorithm and path solution can be developed for an initial seed to the NLP as described in Section 3.4. Transitioning the dynamic equations for the optimal control problem within the barycentric coordinate system is straightforward and is demonstrated in the following scenario.

Optimal Control Problem.

Similar to the two-dimensional approach, each tetrahedron in the defined CSC is solved as a single phase in GPOPS-II and each phase is connected through event constraints. The dynamics constrain the path through each tetrahedron in barycen-

tric coordinates. The optimal control problem formulation has been consolidated as follows.

Minimize the cost functional

$$J = \sum_{p=1}^{P} J^{(p)} \tag{4.26}$$

where

$$J^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} dt \;\; \forall p \in [1...P] \tag{4.27}$$

subject to the dynamic constraints

$$\dot{\alpha}_1^{(p)} = \frac{(EI-FH)\dot{x}^{(p)}-(BI-CH)\dot{y}^{(p)}+(BF-CE)\dot{z}^{(p)}}{det(T_3)} \;\; \forall p \in [1...P] \tag{4.28}$$

$$\dot{\alpha}_2^{(p)} = \frac{-(DI-FG)\dot{x}^{(p)}+(AI-CG)\dot{y}^{(p)}-(AF-CD)\dot{z}^{(p)}}{det(T_3)} \;\; \forall p \in [1...P] \tag{4.29}$$

$$\dot{\alpha}_3^{(p)} = \frac{(DH-EG)\dot{x}^{(p)}-(AH-BG)\dot{y}^{(p)}+(AE-BD)\dot{z}^{(p)}}{det(T_3)} \;\; \forall p \in [1...P] \tag{4.30}$$

$$\dot{\alpha}_4^{(p)} = -\dot{\alpha}_1^{(p)} - \dot{\alpha}_2^{(p)} - \dot{\alpha}_3^{(p)} \;\; \forall p \in [1...P]. \tag{4.31}$$

where A through I is defined by the mapping

$$\mathbf{T} = \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix} = \begin{pmatrix} x_1 - x_4 & x_2 - x_4 & x_3 - x_4 \\ y_1 - y_4 & y_2 - y_4 & y_3 - y_4 \\ z_1 - z_4 & z_2 - z_4 & z_3 - z_4 \end{pmatrix}. \tag{4.32}$$

and the three-dimensional SUAS dynamics,

$$\dot{x}^{(p)} = v\cos(\psi)\cos(\theta) \;\; \forall p \in [1...P] \tag{4.33}$$

$$\dot{y}^{(p)} = v\cos(\psi)\sin(\theta) \;\; \forall p \in [1...P] \tag{4.34}$$

$$\dot{z}^{(p)} = v\sin(\theta) \;\; \forall p \in [1...P]. \tag{4.35}$$

The control is placed on rate of change of the pitch angle, heading angle, and velocity

$$u_1^{(p)}(t) = \dot{\psi}^{(p)}(t) \tag{4.36}$$

$$u_2^{(p)}(t) = \dot{\theta}^{(p)}(t) \tag{4.37}$$

$$u_3^{(p)}(t) = v^{(p)} \tag{4.38}$$

and the state vector defined as

$$X = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \theta, \psi)^T \tag{4.39}$$

with boundary conditions given as the initial and final constraints,

$$\mathbf{X}^{(1)}(t_0^{(1)}) = ((\alpha_1)_0, (\alpha_2)_0, (\alpha_3)_0, (\alpha_4)_0, (\theta)_0, (\psi)_0) \tag{4.40}$$

$$\mathbf{X}^{(P)}(t_f^{(P)}) = ((\alpha_1)_f, (\alpha_2)_f, (\alpha_3)_f, (\alpha_4)_f, (\theta)_f, (\psi)_f). \tag{4.41}$$

Inequality path constraints representing bounds on the state, control and time are defined as

$$0 \leq \alpha_1^{(p)}, \alpha_2^{(p)}, \alpha_3^{(p)}, \alpha_4^{(p)} \leq 1 \tag{4.42}$$

$$|\theta^{(p)}| \leq 180 \ \ deg \tag{4.43}$$

$$|\dot{\theta}^{(p)}| \leq 25 \ \ deg/s \tag{4.44}$$

$$|\psi^{(p)}| \leq 30 \ \ deg \tag{4.45}$$

$$|\dot{\psi}^{(p)}| \leq 10 \ \ deg/s \tag{4.46}$$

$$10ft/s \leq v^{(p)}(t) \leq 30ft/s \tag{4.47}$$

$$0 \leq t^{(p)} \leq \frac{edge_{max}^{(p)}}{v}. \tag{4.48}$$

Finally, event constraints are included to maintain a continuous transition of the state

variables between each phase,

$$X_o^{(p+1)} - X_f^{(p)} = 0 \ \ \forall p \in [1...P-1]. \tag{4.49}$$

The key GPOPS-II parameters are listed below in Table 6.

Table 6. GPOPS-II User Settings, 3D Scenarios

GPOPS-II User Settings	
Mesh Method	hp-PattersonRao
Mesh Tolerance	10^{-3}
NLP Solver	IPOPT
Derivative Supplier	AdiGator
Method	RPM-differential
NLP Tolerance	10^{-5}
Min Collocation Points	4
Max Collocation Points	10
Mesh Fraction	$\frac{1}{2}$*ones(1,2)
Mesh Collocation Points	4*ones(1,4)

4.8 Three-Dimensional Scenario

A simple three constraint model was developed to analyze the three-dimensional scenario. In order to illustrate the effectiveness of the simplex approach, the scenario is solved with previous methods in the literature in a single phase using superellipsoid constraint functions and compared to the solution using a simplex discretization. For the simplex method, a discretization of the space is performed and an A* search algorithm determines the CSC. An initial guess for the path solution is determined by connecting the centroid of each simplex through the CSC. A two-point finite differencing scheme is implemented to acquire initial vectors for the heading and pitch angle, while the heading rate and pitch rate are initiated with the zero vector and the velocity is presented at a maximum value. In order to better compare the two solutions in each formulation, this initial path solution was used to seed the NLP in both the single phase and multiple phase scenarios. The building constraints consists

of three connected polygons, representing a series of buildings along a street with the middle constraint only half the height of the other two as shown in Figure 32.

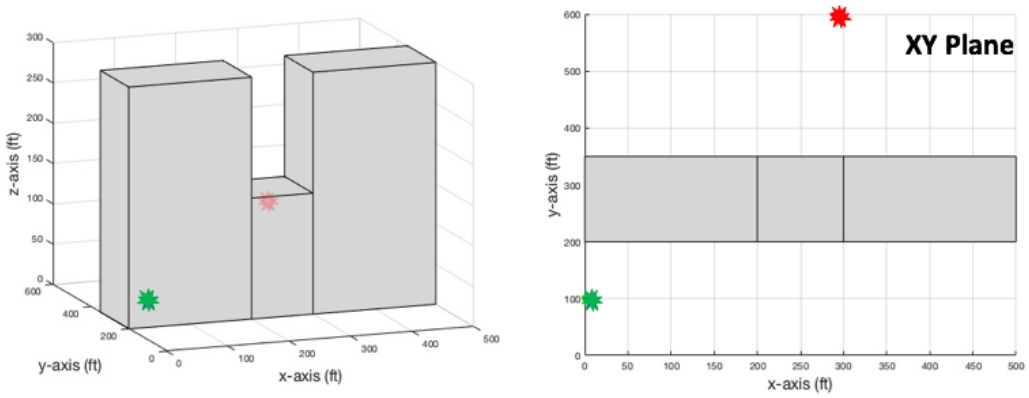

Figure 32. Three-Dimensional Constraint Map

The aircraft begins at level flight flying parallel to the building constraints and is required to climb over the center building and descend to a terminal point perpendicular to the original path. The initial and final aircraft constraints are defined as

$$(x_0, y_0, z_0, \theta_0, \psi_0) = (0, 100, 50, 0, 0) \tag{4.50}$$

$$(x_f, y_f, z_f, \theta_f, \psi_f) = (290, 600, 50, \tfrac{\pi}{2}, 0), \tag{4.51}$$

where the initial and terminal location are illustrated with a green and red asterisk respectfully, as shown in Figure 32.

Single Phase Formulation.

The first formulation illustrates a single phase solution. The problem is solved using Cartesian coordinates with superellipsoid constraint functions implemented in the optimal control solver to model each of the three buildings. The dynamics consist of a five state model consisting of the Cartesian coordinates expressed in Equations

3.15 to 3.17, the heading angle, θ, and the pitch angle, ψ. The control is implemented on the change in heading angle, $\dot{\theta}$, the change in pitch angle, $\dot{\psi}$, and the velocity. The bounds on each of the first three states are consistent with the search domain space and are defined as

$$0 \leq x \leq 500 \tag{4.52}$$

$$0 \leq y \leq 600 \tag{4.53}$$

$$0 \leq z \leq 300. \tag{4.54}$$

The bounds on the remaining states and control parameters are consistent with those defined in Equations 4.43 to 4.48.

The three building constraints are modeled with a superellipsoid function and designed as an inequality path constraint defined as

$$\left(\left(\frac{x_a - x_c}{a_1}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y_a - y_c}{a_2}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z_a - z_c}{a_3}\right)^{\frac{2}{\epsilon_1}} \leq 1. \tag{4.55}$$

Here, the a subscript on the Cartesian coordinates refers to the aircraft position, the c subscript defines the center point of the superellipsoid, while the principle axis in each direction is defined by a_i. The curvature at the edges of the superellipsoid is defined with the ϵ_1 and ϵ_2 term which represent cuboids when they take on values less than one and greater than zero [37]. For this work, both ϵ_1 and ϵ_2 were set to 0.01. Figure 33 shows the superellipsoid shape.

Although the constraint shape looks polygonal, the edges are rounded ever so slightly, creating a small error between the polygonal shape and the superellipsoid. This error is characterized in the results. Finally, the natural log is taken on both sides of Equation 4.55 to minimize the impact the large constraint values can have on the computation time of the NLP solver.

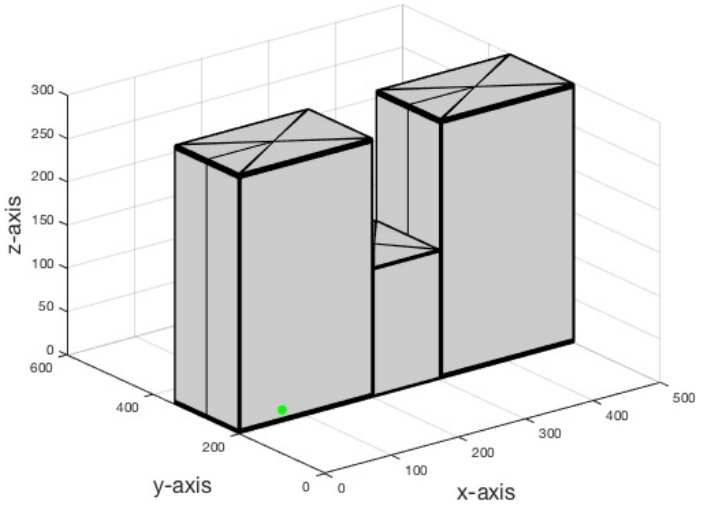

Figure 33. Three-Dimensional Superellipsoid Constraint Map

With the optimal control problem defined for a single phase, an optimal flight path can be computed. Results for the state and control parameters are compared to the results for a CSC solution and are shown in the subsequent sections.

Simplex Formulation.

The new proposed formulation demonstrates the CSC solution and is solved in barycentric coordinates with a phased approach in the optimal control solver. The space is discretized into a tetrahedron set and an A* search algorithm is implemented based on a mid-point heuristic to determine the CSC as described in Section 3.4. The defined CSC is shown in Figure 34.

The dashed red line indicates a path solution, connecting the mid-point of each simplex of the CSC and is used to seed the NLP. Given this defined CSC, the optimal path is computed with a phased approach as defined in the optimal control problem in Section 4.4. These results are compared to the single phase solution in computational time and accuracy.

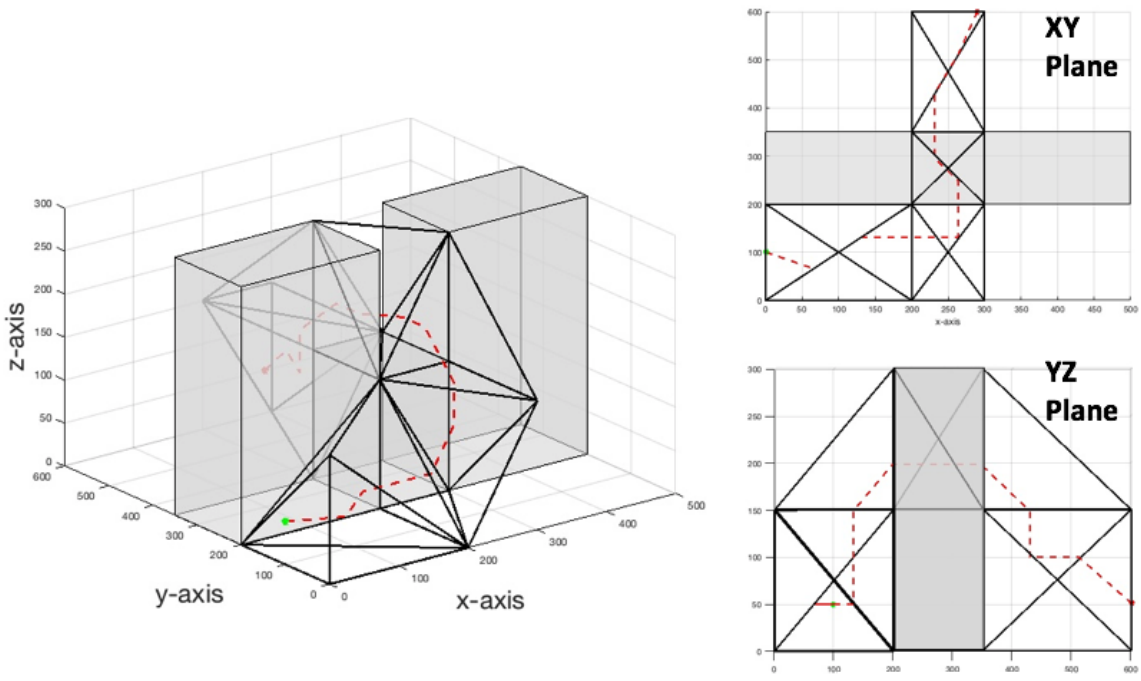

Figure 34. Three-Dimensional Simplex Search Corridor & Initial Guess

Three-Dimensional Results.

The initial path solution used to seed the NLP was determined by connecting the centroid of each adjacent simplex in the defined CSC. In order to draw comparisons between the two formulations, the same path solution was used to seed the NLP in the single phase and multi-phase formulation. This initial path solution can be seen in Figure 35 with the dashed red line. The green asterisks reflect the first scenario in which the optimal solution is solved in one phase within the global domain of the space and satisfies the superellipsoid constraint function. The blue asterisks reflect the second scenario where the optimal solution is determined through each simplex of the defined CSC.

Variation between the two paths can be seen as they climb over the center building with the single phase solution taking a more direct route to the terminal point. The angular state and control parameters are shown in Figure 36.

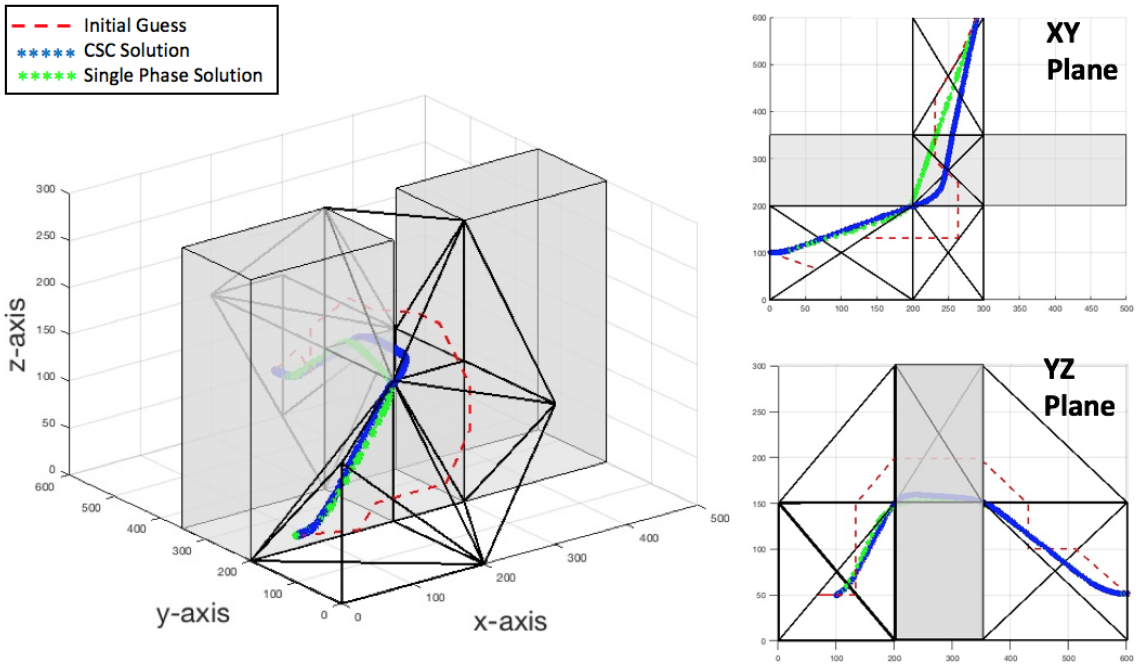

Figure 35. Three-Dimensional Optimal Path Solution Comparison

The top plot in Figure 36 shows the heading angle. As expected with larger tetrahedrons, the centroid solution used to seed the NLP has the most variation in the heading and takes the longest time to complete the path. The single phase and simplex solution only differ in the timing of the turn to fly over the center constraint. The third plot defines the pitch angle with all three path solutions resembling the same angular requirements, however, the computation time of the centroid solution is significantly longer. The second and fourth plots describe the angle rates for the heading and pitch respectively and resemble Pontryagin's principles as expected given the rate limitations on the control variables. Finally, the fifth plot shows the SUAS maintains max speed throughout the simulation in each scenario, however this would not be the case when tighter turn radii are required [83].

The computation and objective times of GPOPS-II are shown in Table 7 for each of the two optimal paths. The objective times between the two scenarios only

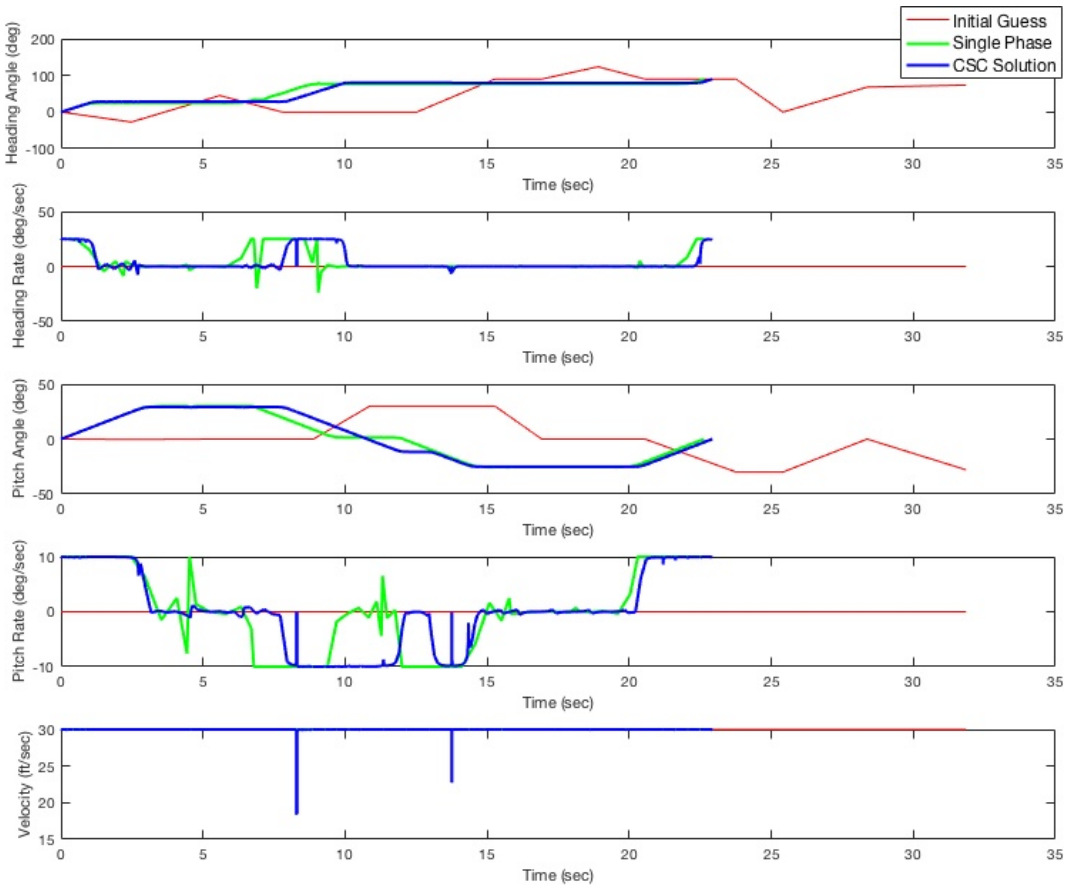

Figure 36. Three-Dimensional Optimal State & Control

Table 7. Three-Dimensional Simulation Results

Solution Type	Comp Time (s)	Obj Time (s)
Single Phase	45.47	22.631
Simplex Solution	5.21	22.77

differ by less than two-tenths of a second while the difference in computation time is significant at over 40 seconds. The disparity in the objective time can be explained in the difference of the constraint function models. Superquadrics were used in modeling the constraints in the single phase solution. The small ϵ_i values increase the sharpness of the constraint edges, but also increases the exponential power of the constraint function. This creates large gradient values within the NLP resulting in a significant increase in computation time. With these epsilon values implemented, there remains

a small error in the shape of the superquadric when compared to the polygonal constraint used in the simplex solution. Characterizing this error is accomplished by reviewing Barr's work for modeling the volume of a superellipsoid [84], defined in his work as

$$V_E = \frac{2}{3} a_1 a_2 a_3 \epsilon_1 \epsilon_2 \beta\left(\frac{\epsilon_1}{2}, \frac{\epsilon_1}{2}\right) \beta\left(\epsilon_2, \frac{\epsilon_2}{2}\right) \tag{4.56}$$

where β represents the beta function and the a_i terms represent the principal axes of the superellipsoid in each direction. The difference between the polygon shape constraint and the superquadric is $1099.6 ft^3$, which results in a 0.012% error. Although this error is small, it is concentrated at the edges of the constraint. The rounded edge of the superquadric function as well as the spacing of the collocation points allows for a more direct flight path over the center constraint. However, the path violates the constraint modeled as a polygon as the path skips over the corner of the constrained edge. The SUAS path solution over the first edge of the center constraint can be seen in the left side and top image in Figure 37.

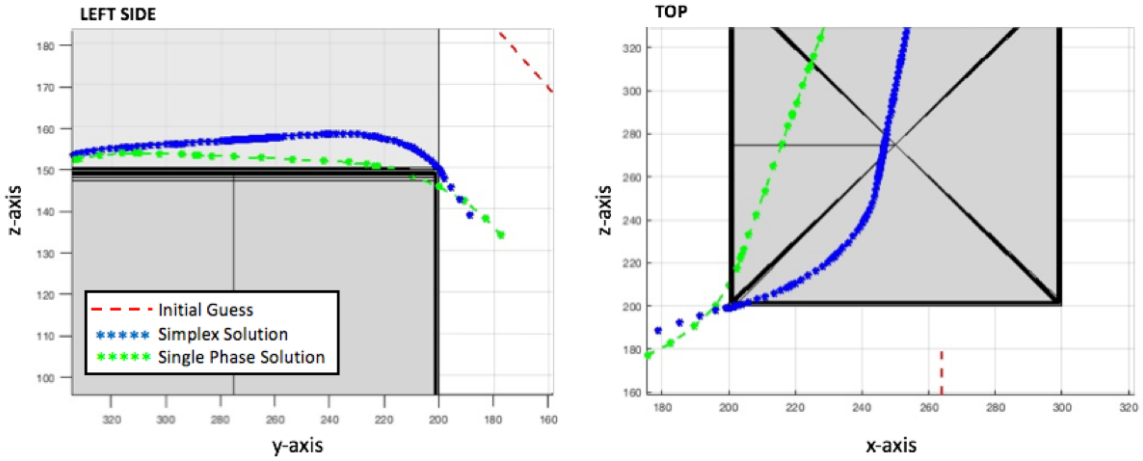

Figure 37. Three-Dimensional Path & Constraint Comparison

The superquadric model is a good representation of a polygonal constraint, how-

ever, to guarantee the true polygonal constraint is not violated, a subjective safety buffer would have to be included into the superquadric model to assure a feasible flight path. By determining a path through a defined CSC, the constraints are eliminated from the domain of the problem and a feasible flight path is presented.

4.9 Three-Dimensional Conclusions

The significant difference between the two and three-dimensional approach for flight through an urban environment is the lack of a readily available three-dimensional discretization and geometric path planning solver. Extending the simplex discretization to three-dimensions, while maintaining the characteristics of a constrained triangulation, is a current research challenge. This work provided a methodology to discretize the constrained domain using simple rectangular prism shapes. Additionally, fast geometric path planners used in the two-dimensional analysis contain an A* search algorithm as well as a funnel algorithm based on a heuristic that has been tuned and developed for the specified path solution. The approach taken in the three-dimensional scenario implements a straightforward A* search with a centroid heuristic to attain a rudimentary path solution. Both the discretization and initial path generation can be significantly improved to allow for more complex constraint environments and a more accurate path solution to seed the NLP of the optimal control solver.

Given a discretization of the space and an initial path solution, extending the principles of the two-dimensional optimal control problem to three-dimensions is straightforward under the CSC construct. Three-dimensional vehicle dynamics can be implemented with barycentric coordinates and a path solution can be attained by solving the path one simplex at a time and connecting each simplex solution through path constraints in the optimal solver. The results attained in this work illustrate

the benefits that can be achieved by formulating the problem with a CSC approach resulting in strictly defined parameter bounds where the constraints imposed by the urban infrastructure are eliminated from the search space of the NLP solver. Future work developing efficient three-dimension discretization and geometric search algorithms will further increase computational speed and accuracy and allow for rapid solutions to more realistic scenarios.

V. Variations to the Two-Dimensional Problem

5.1 Overview

Chapter V presents multiple variations to the two-dimensional constrained optimal control problem. Chapter IV successfully demonstrated the simplex methodologies for a realistic, two-dimensional scenario. In this chapter, speed control is added to the optimal control problem to allow variation in the SUAS turn radius, which expands the feasible search space of the domain. First, the SUAS is required to visit multiple waypoints incorporated into the Chicago city map, significantly increasing the number of phases in the problem and demonstrating the ability to reach narrow corridor regions. Second, keep-out regions are implemented, demonstrating a scenario where the aircraft is required to minimize incursion though undesirable airspace. Next, constant wind fields are incorporated into the model to demonstrate the feasibility and issues presented when exogenous inputs are included into the CSC construct. Finally, contingency planning is evaluated for the scenario in which a simplex search corridor becomes obstructed mid-way through a flight plan. Each of these scenarios are evaluated with minimal changes to the optimal control problem and demonstrate the flexibility and utility of the CSC method.

5.2 Optimal Control Problem

For each scenario evaluated in this chapter, the objectives, parameters, rates, limits, and bounds of the optimal control problem are defined below. Small changes are necessary dependent on the specific scenario which is detailed in the appropriate section.

The objective is to minimize the time of flight through the scenario, defined as

$$J = \sum_{p=1}^{P} J^{(p)} \tag{5.1}$$

where

$$J^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} dt \;\; \forall p \in [1...P]. \tag{5.2}$$

The state dynamics include

$$\dot{\alpha}_1^{(p)}(t) = \frac{(y_2 - y_3)v(t)\cos\theta(t) + (x_3 - x_2)v(t)\sin\theta(t)}{det(T)} \tag{5.3}$$

$$\dot{\alpha}_2^{(p)}(t) = \frac{(y_3 - y_1)v(t)\cos\theta(t) + (x_1 - x_3)v(t)\sin\theta(t)}{det(T)} \tag{5.4}$$

$$\dot{\alpha}_3^{(p)}(t) = -\dot{\alpha}_1^{(p)}(t) - \dot{\alpha}_2^{(p)}(t) \tag{5.5}$$

$\forall p \in [1...P]$, resulting in the state vector

$$\mathbf{X} = (\alpha_{\mathbf{1}}, \alpha_{\mathbf{2}}, \alpha_{\mathbf{3}}, \theta, \dot{\theta}, \mathbf{v}). \tag{5.6}$$

The control for the SUAS is on the change in heading rate and acceleration resulting in

$$u_1^{(p)}(t) = \ddot{\theta}^{(p)}(t) \tag{5.7}$$

$$u_2^{(p)}(t) = a^{(p)}(t) \tag{5.8}$$

Bounds are applied on the states, control, and time to limit the search domain of the NLP solver. The bounds on the first three position states are enforced with the start

and end location of each interval

$$[\alpha_1, \alpha_2, \alpha_3]^{(1)} = start\_position \tag{5.9}$$

$$[\alpha_1, \alpha_2, \alpha_3]^{(P)} = end\_position \tag{5.10}$$

with the phases for the first three states bounded by

$$0 \leq \alpha_1^{(p)}, \alpha_2^{(p)}, \alpha_3^{(p)} \leq 1 \tag{5.11}$$

$\forall p \in [2...P-1]$. The bounds for the remaining states, control, and time within each phase are enforced as

$$|\theta^{(p)}| \leq \pi \tag{5.12}$$

$$|\dot{\theta}^{(p)}| \leq 25 \ \ deg/s \tag{5.13}$$

$$|\gamma^{(p)}| \leq 5 \ \ deg/s^2 \tag{5.14}$$

$$10 \ \ ft/s \leq v^{(p)} \leq 30 \ \ ft/s \tag{5.15}$$

$$|a^{(p)}| \leq 2 \ \ deg/s^2 \tag{5.16}$$

$$0 \leq t^{(p)} \leq \rho \frac{edge_{max}^{(p)}}{v} \tag{5.17}$$

$\forall p \in [1...P]$ and where $edge_{max}$ represents the longest edge in the defined simplex and ρ is a scaling factor to allow for additional time spent in a simplex due to exegonous inputs or keep-out regions.

Finally, the event constraints required for the continuous transition of the state and control across each phase boundary are defined by

$$X_o^{(p)} - X_f^{(p-1)} = 0 \ \ \forall p \in [2...P]. \tag{5.18}$$

5.3 Waypoint Following

Waypoint Scenario Development.

The scenario presented considers the two-dimensional flight of a SUAS through an urban environment modeled in downtown Chicago, USA. Buildings that reach an altitude greater than 550 AGL are modeled as polygonal constraints. Given a starting location, the aircraft is required to fly over three separate waypoints before returning to the start location. This results in a four interval solution. For the work herein, the required start and waypoint locations are defined as

$$start = [200, 200]ft \tag{5.19}$$

$$waypoint_1 = [1825, 1700]ft \tag{5.20}$$

$$waypoint_2 = [2370, 2355]ft \tag{5.21}$$

$$waypoint_3 = [3650, 1215]ft. \tag{5.22}$$

The aircraft is required to maintain an altitude of 600 feet AGL while avoiding all building constraints. The Triplanner algorithm is initiated with the start and end location for each interval as well as the minimum turn radius, R, defined by the vehicles minimum velocity and turn rate limit.

$$R = \frac{v_{min}}{\dot{\theta}_{max}} \tag{5.23}$$

The SUAS maintains control on the change in heading rate and acceleration, thus allowing varying rate turns and the potential to maximize the search domain. The waypoints were chosen to illustrate the vehicle control and provide for a challenging optimal control problem. The first waypoint is located on the front doorstep of the associated building, requiring the SUAS to fly adjacent to the constrained edge

followed by a sharp turn to the north of the city. The second waypoint monitors the center of an intersection surrounded by four constraints on each corner. Finally, the third waypoint is located at a metro station designed to extend the four interval problem through a large number of simplexes. Each waypoint is shown with a black dot in Figure 38.

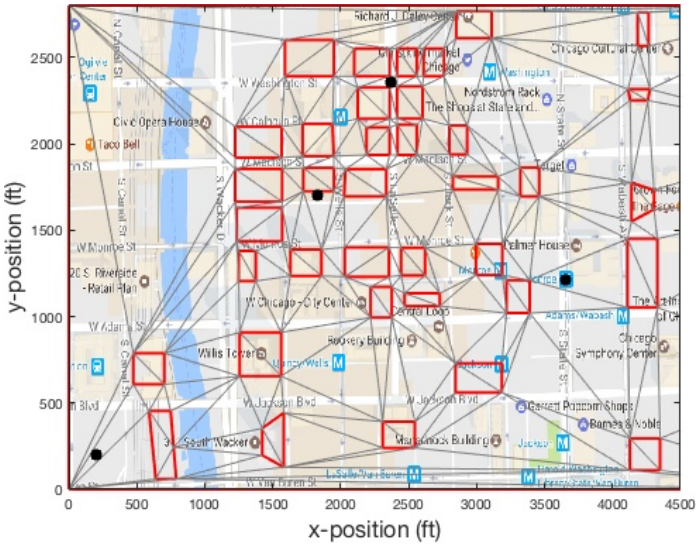

Figure 38. Multiple Waypoint Constraint Map. [Map Data @2017 Google]

When the SUAS is required to visit multiple waypoints, the Triplanner algorithm can be used to generate an initial guess for each interval, defined as the path between consecutive waypoints. Within each interval, a phased approach is used where each simplex represents an individual phase. As a new interval is introduced, the first phase will be identical to the last phase of the previous interval. This process is shown below in Figure 39 with δ_i defining the current interval.

As the path solution terminates at a waypoint, represented at phase P in interval δ_i, the following interval begins its path at the same waypoint in the same phase, now designated as phase 1 in interval δ_{i+1}. The overlap of this phase is essential to assure a continuous transition of the optimal control problem. The interval structure for the bounds, mesh, initial guess, and events are then combined into a single structure with

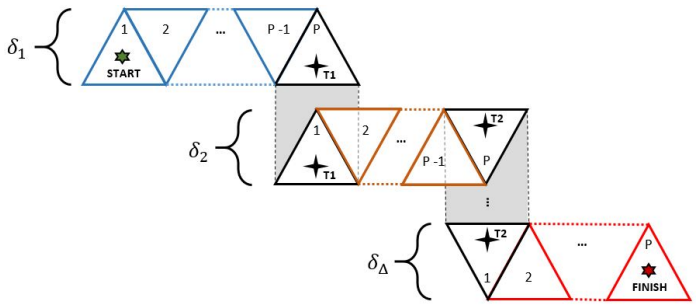

Figure 39. Multiple Waypoint Interval Development

the number of phases, P_t, equal to the sum of the phases in each waypoint interval defined by

$$p_t = [p^{\delta_1}; p^{\delta_2}; \ldots; p^{\delta_\Delta}] \tag{5.24}$$

$\forall p \in [1...P]$ and $p_t \in [1...P_t]$.

To assure each waypoint is visited within a single optimal control problem, a continuous transition of the states is required at each interval. An event constraint is added to the optimal control problem defined in Section 5.2 defined by

$$X_o^{(\delta)} - X_f^{(\delta-1)} = 0 \;\; \forall \delta \in [2...\Delta] \tag{5.25}$$

where δ represents the current interval and Δ defines the total number of intervals in the solution.

Finally, the objective of this optimal control problem is to solve the minimum flight path over all phases subject to the dynamic constraints defined in Equations 5.3-5.5, parameter bounds of Equations 5.9-5.17, and event constraints of Equations 5.18 and 5.25. The cost function of Section 5.2 is replaced with one which accounts

for flight across all intervals defined as

$$J = \sum_{p_t=1}^{P_t} J^{(p_t)} \tag{5.26}$$

where

$$J^{(p_t)} = \int_{t_0^{(p_t)}}^{t_f^{(p_t)}} dt \;\; \forall p_t \in [1...P_t]. \tag{5.27}$$

$$\tag{5.28}$$

Waypoint Scenario Results.

The Triplanner algorithm is used to determine four individual path solutions, representing each defined interval. Figure 40 shows the connected four interval solution and the defined CSC.

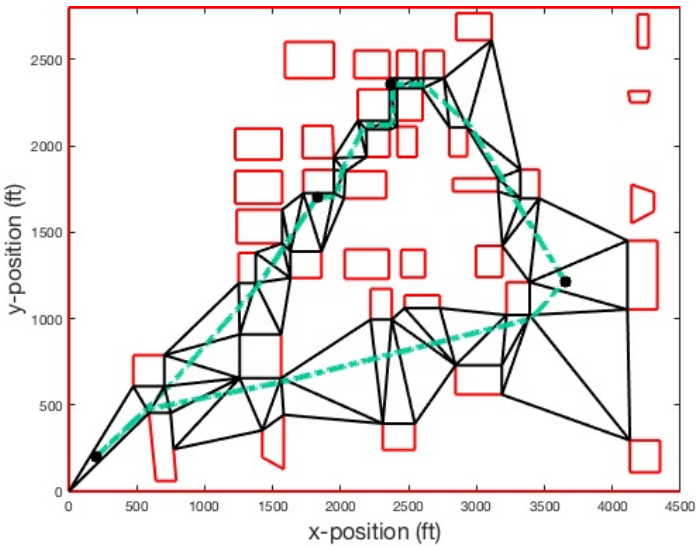

Figure 40. Multiple Waypoint Triplanner Solution

This geometric path solution is shown with the dashed green line and avoids the polygonal path constraints with a radial distance calculated from the minimum flight speed as defined in Equation 5.23. The path contained within each interval results in

a sub-optimal Dubins path solution as the minimum radius turns are not optimally located due to the constraint field and the formulation of the Triplanner algorithm. Additionally, the heading rate limits are not upheld at the waypoint locations resulting in a discontinuity in the vehicle heading angle. The CSC for each interval is shown with the black outlined polygons and are completely maintained outside of all path constraints. This provides a CSC for the optimal control solver free of all building constraints. The affects of Triplanner's greedy A* search algorithm can be seen in the second interval as the path chosen requires two turns around the last building constraint before the second waypoint vice a shorter path containing only one turn. The optimal solution will be determined within this CSC and therefore may not be the global optimal solution. The optimal CSC could be determined by increasing the search time of the A* algorithm, but at the cost of computation time. Table 8 shows the number of phases and computation time for each of the intervals of the Triplanner solution.

Table 8. Triplanner Interval Solutions

Interval	Phases	Computation Time (ms)
1	17	7.37
2	15	5.96
3	18	4.78
4	23	6.89

The combined path results in 73 phases. The path solution for Triplanner is non-smooth at each waypoint and therefore the path length is not recorded as it cannot be directly compared to the optimal solution.

The initial guess for the optimal control solver can now be formed by combining the Triplanner solutions of the four intervals into appropriately formatted structures. Figure 41 shows the optimal path contained within the defined CSC.

The optimal path is color coded with blue and dark green asterisks to illustrate the

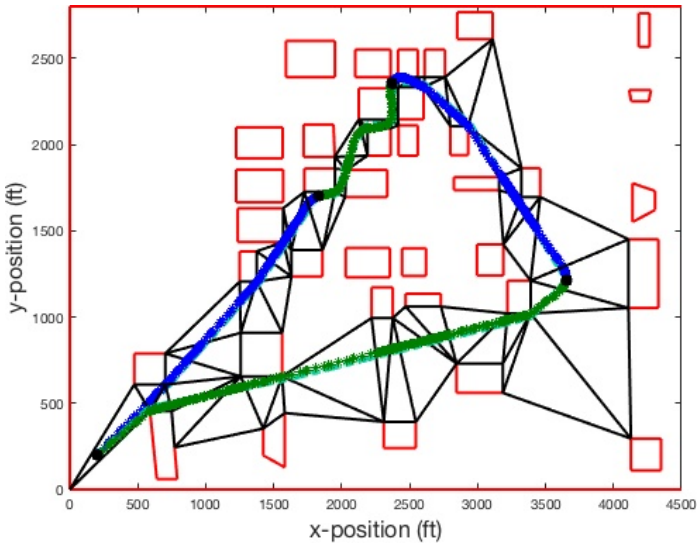

Figure 41. Multiple Waypoint Hybrid Solution

different intervals of the problem. The transition between intervals is continuous and rate limited according to the limitation of the problem state and control variables and enforced through the event constraints of the optimal control problem described in Equations 5.18 and 5.25. The path solution is optimized over the Dubins path result from Triplanner as the turn points are moved to an optimal location around the building constraints allowing for a more direct path to the next waypoint. The final path solution is comprised of 73 phases with an objective time of 290.51 seconds. The required computation time for the optimal solution in GPOPS-II was 117.45 seconds on a PC. This time was extended significantly due to the discontinuities in the Triplanner solution as well as the extended number of required phases. The state vectors for the heading, heading rate, and velocity as well as the control vectors are shown in Figure 42.

The initial guess, formed from the result of the Triplanner solution, is shown with the dashed green line while the optimal state and control vectors are shown with the solid blue lines. Evaluating the heading angle in the first plot, the discontinuities in the Triplanner heading angle can be seen at each waypoint, located at 74, 116,

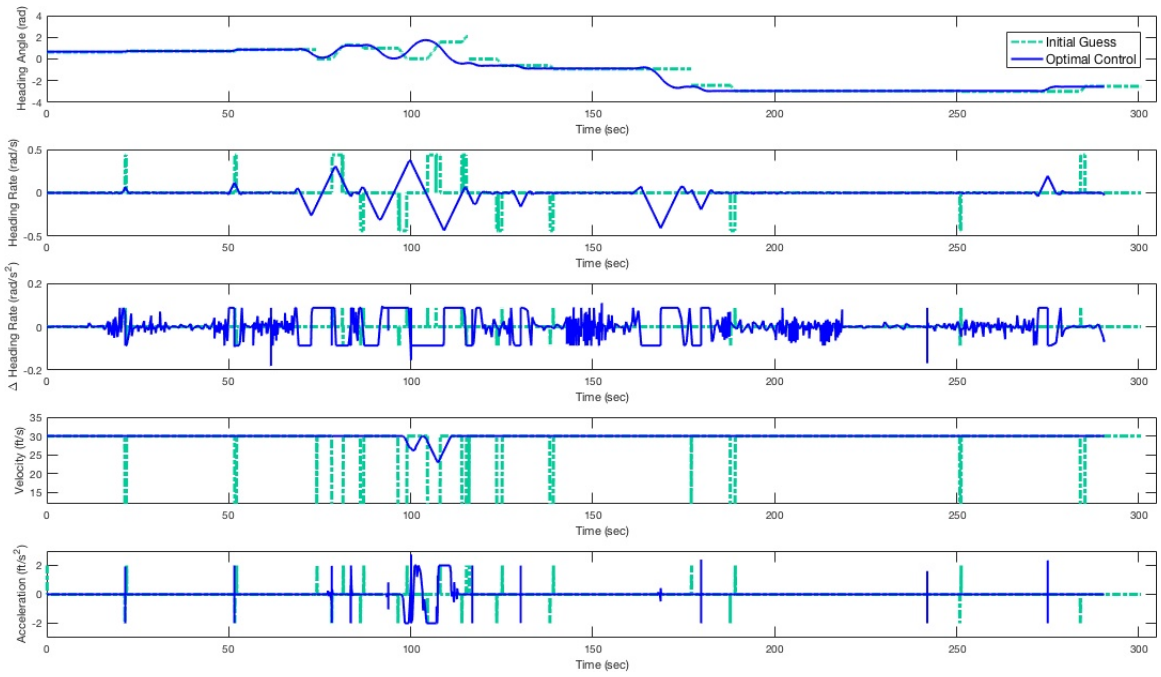

Figure 42. Multiple Waypoint Hybrid Solution State & Control

and 177 seconds respectively. The optimal heading rate resembles the turn points of the Triplanner solution, but is solved slightly faster than the initial guess provided. The second plot of Figure 42 shows the heading rate. The initial guess provided maximum rate turns based on the Dubins path solution of Triplanner. The optimal heading rate was limited on the vehicles turns due to the rate-limited control vector shown in the third plot. Finally, the velocity and acceleration vectors are shown in plots four and five respectively. The control on the vehicle's acceleration is illustrated in the velocity vector as the vehicle is required to slow down in order to satisfy turn rate limits designed at the second waypoint.

Waypoint Scenario Conclusions.

The fast computation times of these geometric solutions provide a foundation for achieving real-time, onboard operations with optimal control software. The Triplanner algorithm successfully provided a discretized triangular mesh, a CSC free of path

constraints, and an initial solution for the SUAS position states. However, since Triplanner provides a point to point solution, the intervals were solved individually creating a discontinuity in the heading angle of the estimated solution. This creates the requirement for optimal control software as the 2010 version of the Triplanner algorithm does not account for initial or final heading angle constraints or rate limited control inputs. The initial path solution was evaluated and heading angle rates, velocity, and acceleration vectors were established. These vectors were implemented in the optimal control solver, GPOPS-II for calculation of the optimal path within the desired CSC.

By defining the problem in barycentric coordinates, implementation of the Triplanner solution as the initial guess was accomplished through a phased approach where each simplex represented a single phase of the optimal solution. A continuous transition of the states and control were accomplished through event constraints, allowing the optimal solution to continuously transition from one interval to the next across each waypoint. Further, the challenging urban environment presented tight corridors between building constraints where the vehicle was required to slow its airspeed in order to reduce the minimum turning radius required to achieve the desired path. By providing control to the vehicle's acceleration, minimal speed deviations were realized and a continuous minimum time solution was achieved over four intervals containing three waypoints. The computation times for the optimal control problem exceeded the limits for real-time operations, but the simulation presented provides a foundation for future work where computational times could be drastically reduced by smoothing the discontinuities of the Triplanner solution through a low-pass filter or determining the appropriate limits for a series of finite-horizon optimal control problems combined to meet the same objective posed herein.

5.4 Aircraft Keep-Out Regions

Keep-Out Region Scenario Development.

Consider a city map representing an urban environment, where an aircraft is required to maintain constant altitude and fly from an initial position to a terminal point. As before, this scenario is represented with a constraint map of downtown Chicago, USA. The SUAS is required to maintain an altitude of 600 ft AGL. Each building that exceeds 550 ft AGL is modeled as a constraint that must be avoided. Keep-out regions representing minimal flight zones are included along the West river and down State Street. These zones must be avoided when possible. Figure 43 illustrates the constraint map for the intended scenario as well as the triangulated mesh over the domain.

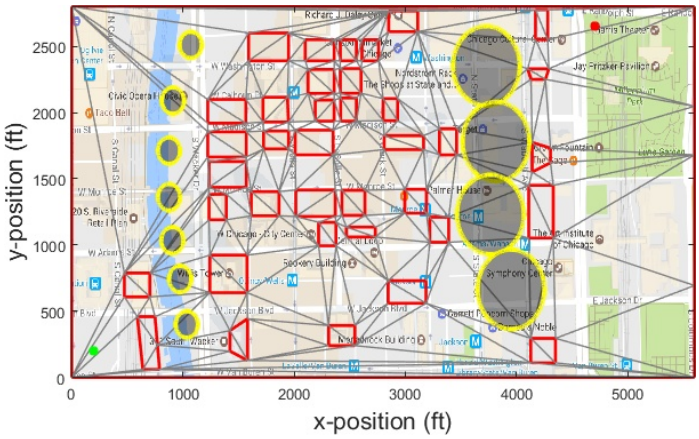

Figure 43. Keep-Out Region Constraint Map. [Map Data @2017 Google]

The initial starting position and the target location are shown with the green and red asterisk respectively. The depicted red polygons outline building constraints that must be avoided while the grey keep-out regions are modeled as circular regions upon which the SUAS must minimize incursions. The left column of keep-out regions have a separation between them, allowing the aircraft to fully avoid the flight zones if they reside completely inside the defined CSC. The right column of keep-out regions

maintain a small overlap, requiring flight through the zones at a minimum incursion level, dependent on the CSC boundaries. Other locations and sizes for keep-out regions may require the aircraft to fly through the center of the undesirable airspace. If a keep-out region is unavoidable, an analysis on the threat should be conducted and portions of the threat region should be modeled as a hard constraint and removed from the search space if warrented.

With the dynamics and parameter bounds defined previously in Section 5.2, the newly defined objective function is designed for minimum time of flight, while including penalties for incursions into the keep-out regions. The cost associated with the minimum time flight through each phase is represented as

$$J_{minT}^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} dt \;\; \forall p \in [1...P]. \tag{5.29}$$

The penalty for the keep-out regions, $F_i(x, y)$ is defined with a sigmoid function, $\phi(F)$, and minimizes the keep-out incursions at each collocation point as follows

$$\phi_i(F_i) = \frac{1}{1+e^{(s_i(F_i(x,y)-1)}}, \tag{5.30}$$

for

$$F_i(x, y) = \left(\frac{(x^{(p)}(t)-Kx_i)}{a_i}\right)^2 + \left(\frac{(y^{(p)}(t)-Ky_i)}{b_i}\right)^2 \tag{5.31}$$

where Kx_i and Ky_i define the keep-out center point and a_i and b_i define the semi-major and semi-minor axes. This yields a cost function, minimizing the incursion to keep-out regions, defined as

$$J_{minE}^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} \phi_i^{(p)}(F_i)dt \;\; \forall p \in [1...P-1] \tag{5.32}$$

The complete objective function is a summation of the minimum time cost and minimum incursion to keep-out regions.

$$J = \sum_{p=1}^{P} \beta J_{minT}^{(p)} + \sum_{p=1}^{P-1} (1-\beta) J_{minE}^{(p)} \tag{5.33}$$

Here, β defines the weighting on the components of the cost such that the cost function influences the desired flight path.

Keep-Out Region Scenario Results.

The initial guess used to seed the NLP is determined using the Triplanner algorithm. The output of Triplanner, shown in Figure 44, consists of a Dubins path solution contained inside a CSC shown as a series of black simplexes.

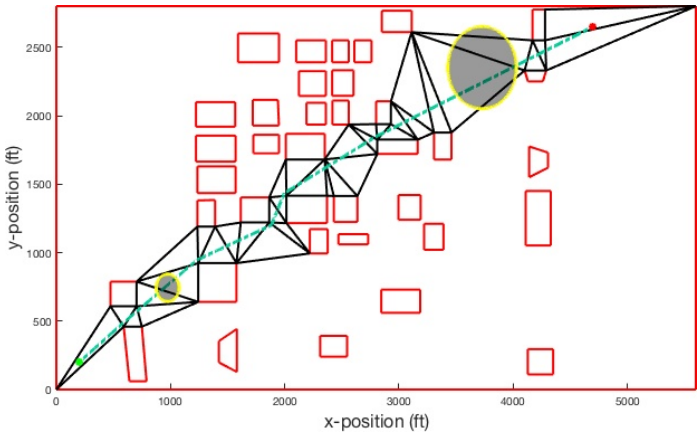

Figure 44. Keep-Out Region Triplanner Solution

All building constraints are contained outside of the CSC thus eliminating path constraints from the problem formulation. The initial path found by Triplanner is illustrated with the green dashed line and does not account for keep-out regions but maintains a clearance from each building equal to the minimum turning radius of the SUAS. Note that only the simplexes in the CSC, as well as the sigmoid functions modeling the keep-out regions, are presented to the NLP solver so that a computa-

tionally efficient search can be performed for the optimal solution within the defined CSC.

Figure 45 shows the resultant optimal control solution given the initial guess from Figure 44 and the user settings defined in Table 3.

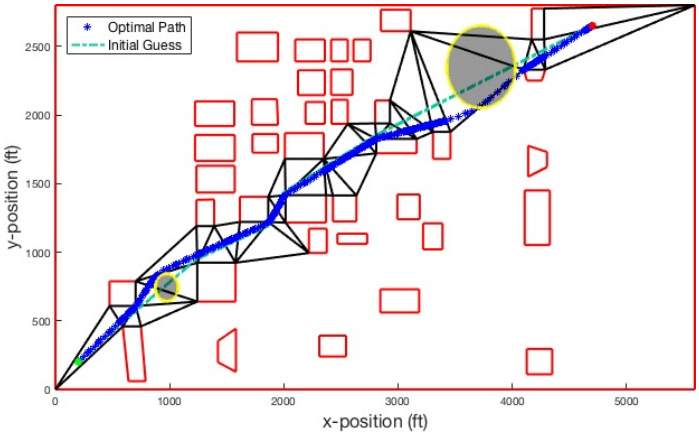

Figure 45. Keep-Out Region Hybrid Solution

The collocated points of the optimal path through the CSC are shown with the blue asterisks. The first keep-out region is fully avoided since the region is located completely within the CSC. The second keep-out region extends beyond both boundaries of the CSC and therefore the path is dependent on the weighted cost function, set to $\beta = 0.7$ in this work. The optimal path chosen avoids the keep-out region up to the limit of the CSC. Due to the barycentric problem formulation, the search space is restricted to the bounds of the current simplex and therefore must remain inside the defined CSC and incur a penalty to the cost function when inside the keep-out region. If this incursion exceeds a mission threat level, the keep-out region should be modeled as a hard building constraint and removed from the search space.

The vehicle heading and heading rate vectors are shown in the top two plots of Figure 46. The third plot shows the integrated value of the cost function of Equation 5.32 evaluated over each phase. Finally, the remaining three plots show the change

in heading rate, velocity, and acceleration respectively.

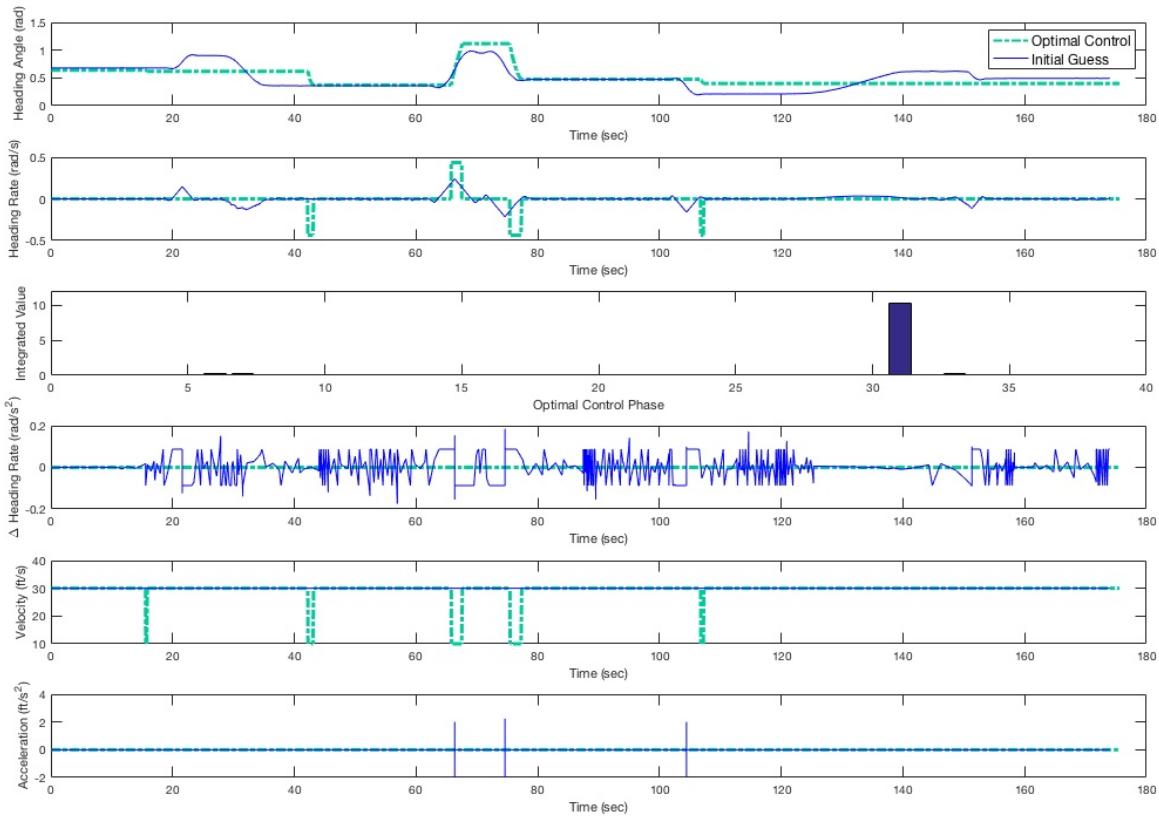

Figure 46. Keep-Out Region Hybrid Solution State & Control

The deviation in the heading for the first keep-out region can be seen in the heading vector between 20 - 40 seconds into the flight. The lower heading value at the 70 second mark illustrates the benefits gained from the optimal solution over the Dubins path solution. The second keep-out region is illustrated at the 105 second mark of the first plot. The integrated value of the sigmoid function on the third plot shows the impact the two keep-out regions have on the optimal control solution. The integrated values are referenced in each phase rather than by flight time as the cost associated with the sigmoid function corresponds to individually defined simplexes and the time associated with each simplex is not consistent, as the size of the simplexes are dependent on the discretized mesh and therefore not equally

spaced. The first keep-out region only slightly affected the cost function as seen in phase six and seven. The second keep-out region is shown to have a greater affect on the cost function, as the region is unavoidable, shown in phases thirty-one and thirty-three. The value of the constraint portion of this cost can be adjusted through the stiffness parameter of the sigmoid function described previously. With a value of $s = 2$, a smooth, differential function is evaluated, but impacts to the cost function can be seen when the vehicle approaches a close proximity to the region, while not necessarily entering the region. Finally, through the weight parameter of the total cost function, β, the impact of entering a keep-out region can be adjusted.

Keep-Out Region Conclusions.

The Triplanner algorithm is a computationally efficient algorithm that provides a quality guess and a defined CSC to the NLP solver allowing for the avoidance of hard constraints. In designing the optimal control problem, constraints must be evaluated to determine if they consist of a flight region for which incursions can be minimized, or if it is a constraint that must be avoided completely. If the constraint must be completely avoided, the Triplanner algorithm's triangulated mesh will remove the constraint from the search field, limiting the problem's domain. Multiple constraints, combined with the minimum turning radius of the SUAS, could result in Triplanner failing to return a feasible path solution. If however, constraints are modeled as keep-out regions where the SUAS must minimize time spent in a flight zone, the constraint should be modeled in the optimal control problem rather than the Triplanner algorithm. By modeling the constraint in the cost function, tuning parameters can be used for varying levels of incursions within the keep-out region based on the defined CSC and the risk of designing an over-constrained problem is reduced.

The sigmoid function was shown to be a viable option for modeling constraints

in the optimal control cost function within the construct of a CSC. These functions are appropriate for gradient-based optimization software as they provide smooth, bounded, and differentiable functions. The final cost function was a weighted sum, distributing the cost over flight time and time within keep-out regions. These weights, along with the stiffness parameter of the sigmoid function can be tuned to achieve desired results based on the level of incursion permitted within the keep-out regions.

Ultimately, a discretized simplex mesh was used to provide a foundation for optimal control solutions. When paired with direct orthogonal collocation methods for optimal control, both hard constraints, such as buildings and terrain, as well as keep-out regions, such as unavoidable flight zones, can be modeled and optimal path solutions can be attained.

5.5 Wind Analysis

Wind Analysis Algorithm Development.

This section demonstrates the implementation of a constant wind field into the simplex model. Although a constant wind field may not be realistic in the urban environment, this scenario illustrates the feasibility of the approach, outlines a few of the issues that arise when incorporating wind parameters, and sets a foundation for future work modeling dynamic wind fields. Further, as stated in Chapter I, all wind fields for this scenario are constant and assumed to be know a priori. This work is accomplished in the open loop without regard to feedback on system parameters and therefore yields the flight path feasibility based on the expected wind environment.

The wind field is added to the dynamics of the aircraft and then embedded into the barycentric coordinates of the simplex structure. The resulting dynamic equations replace Equations 5.3 and 5.4 in the previously defined optimal control problem and

become

$$\dot{\alpha}_1^{(p)}(t) = \frac{(y_2-y_3)v(t)[\cos\theta(t)-\beta cos\chi]+(x_3-x_2)v(t)[\sin\theta(t)-\beta sin\chi]}{det(T)} \tag{5.34}$$

$$\dot{\alpha}_2^{(p)}(t) = \frac{(y_3-y_1)v(t)[\cos\theta(t)-\beta cos\chi]+(x_1-x_3)v(t)[\sin\theta(t)-\beta sin\chi]}{det(T)} \tag{5.35}$$

where β defines the wind magnitude and χ defines the wind direction. The wind direction is defined by a westerly wind when $\chi = 0$. All remaining state, controls, bounds and objectives of the optimal control problem defined in Section 5.2 remain unchanged.

Wind Analysis Scenarios Results.

Two scenarios were developed to illustrate the inclusion of constant wind parameters within the simplex construct for the optimal control problem. For each scenario, the SUAS is required to minimize time from an initial starting point to a final point, defined as

$$start = [200, 200]ft \tag{5.36}$$

$$finish = [4700, 2650]ft. \tag{5.37}$$

The first scenario varies the wind magnitude from $\beta = 0$ to $\beta = 8ft/s$. The direction of the wind is held constant at $\chi = 90^o$. Figure 47 shows the path for each defined magnitude of the wind. The highlighted region in this figure illustrates a portion of the CSC where the heading angle is required to change significantly through the path. Here, the increased difficulty for the NLP solver can be seen as the aircraft encounters increasingly larger magnitudes of wind coming from the south. Limitations on the path development exist first because of the constrained CSC path and second because of the NLP's dependency on the initial seed provided to the NLP solver. Within the

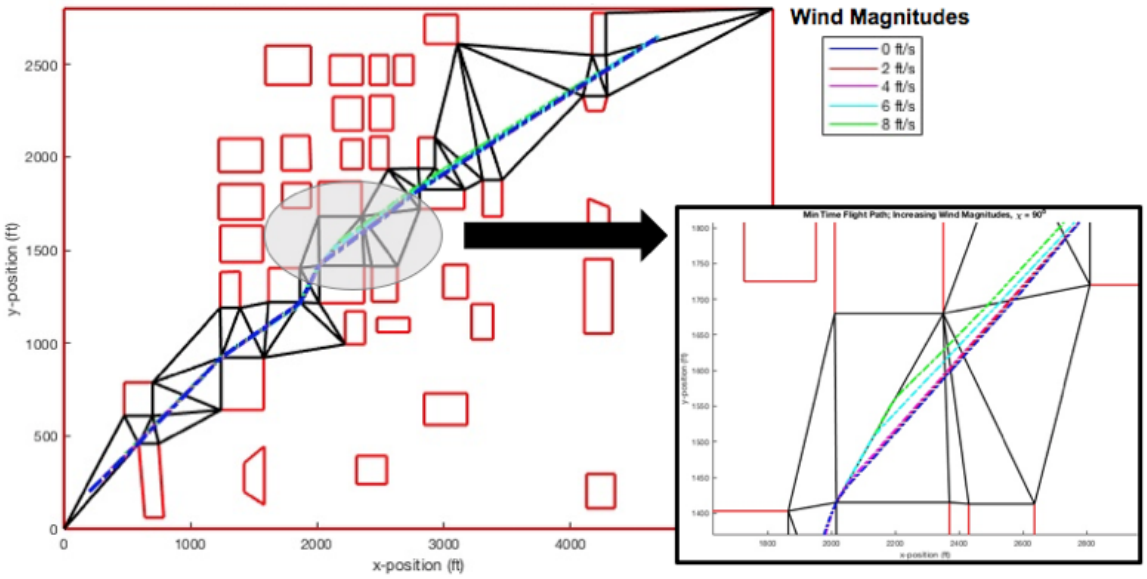

Figure 47. Min Time Flight with Varying Wind Magnitude

simplex construct, as the path traverses around a vertex point, there are simplex regions that are only occupied by the path for very short durations as the path only crosses the vertex point. Therefore, the mission time and collocation points in this phase of the optimal control problem are minimal in the initial seed and impact the resulting NLP solution. In Figure 47, the aircraft's path is extended along the boundary edge of the simplex as the wind magnitude is increased. This is a result of both of these limitations. Figure 48 describes the heading and heading rate for each of these paths. Here the turning points for the vehicle are dispersed in accordance with the wind magnitude.

The second scenario held the magnitude of the wind constant at $\beta = 7ft/s$ while the direction of the wind was varied over 360 degrees in 45 degree increments. Figure 49 shows the path for each defined magnitude of the wind. Again, the limitations from the previous scenario are shown as the path can be seen aligning to the edge of simplex as the wind becomes more northerly. Figure 50 describes the heading and heading rate for each of these paths in the second scenario. The turn points can be

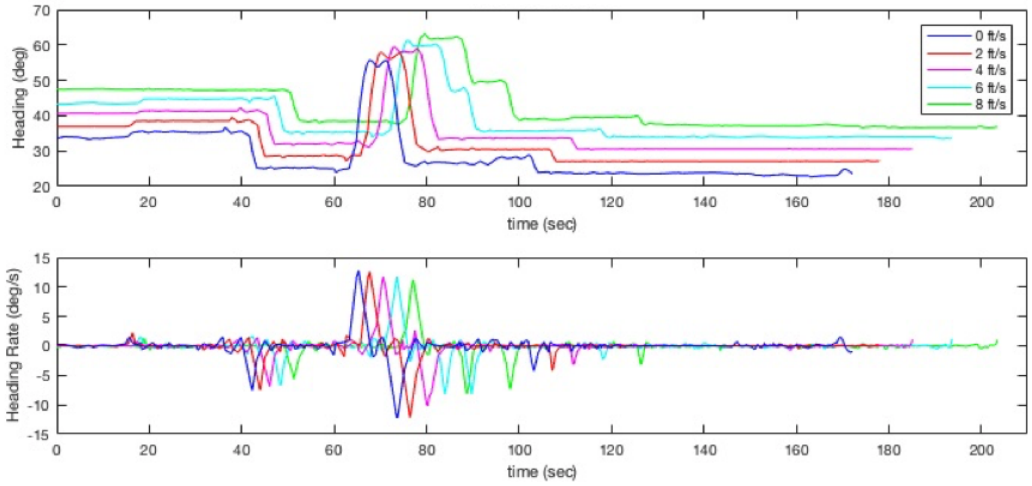

Figure 48. Varying Wind Magnitude, State Trajectories

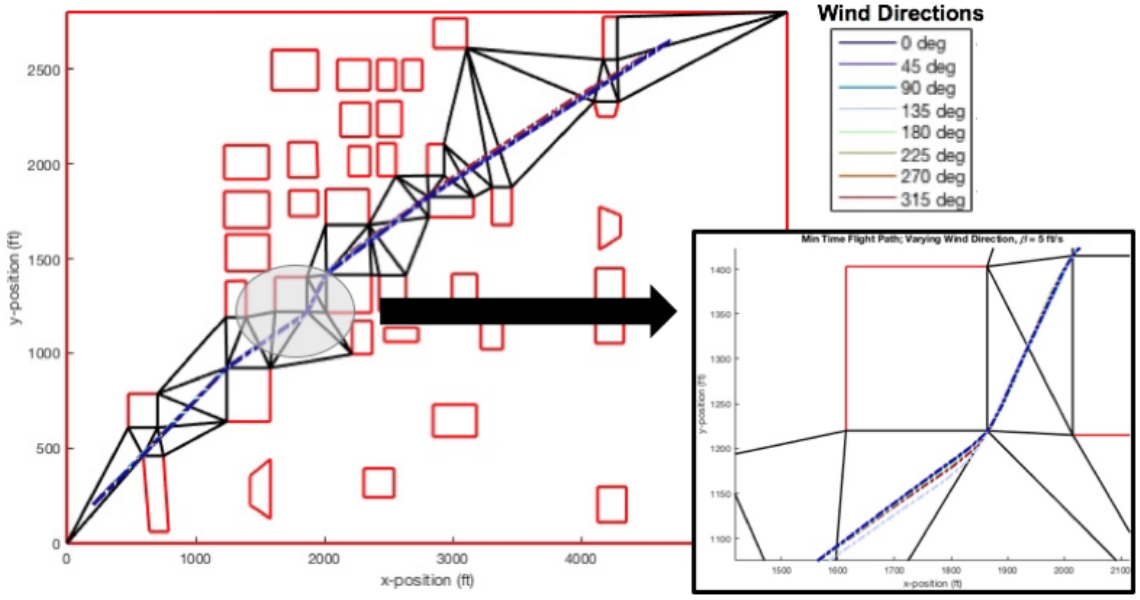

Figure 49. Min Time of Flight with Varying Wind Direction

seen to vary based on the wind direction, with the darker lines representing easterly winds and the lighter lines representing winds from the west.

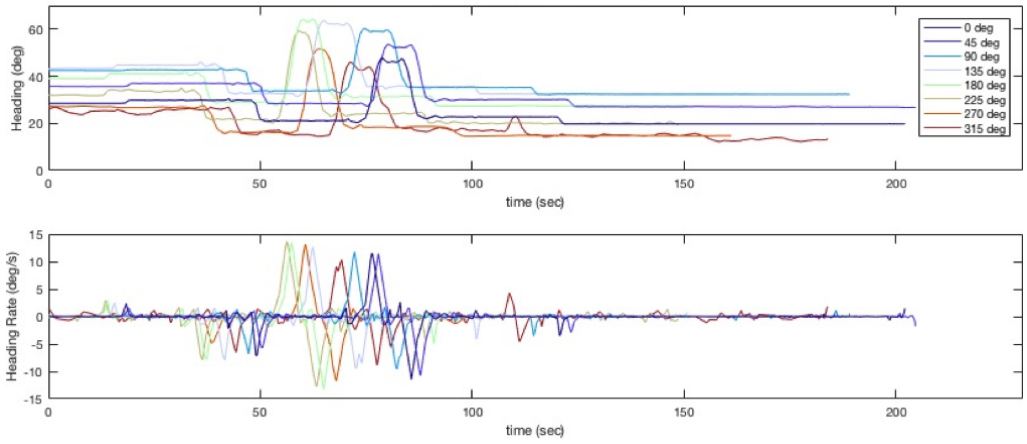

Figure 50. Varying Wind Direction, State Trajectories

Wind Analysis Conclusions.

These two scenarios successfully showed the simplex flight path with constant winds incorporated into the dynamics while varying magnitude and direction. However, with the aircraft's max speed of $30 ft/s$, the NLP solver had difficulties converging on a solution for wind fields greater than $8 ft/s$. This is a result of the flight path being restricted to the defined CSC, where narrow corridors weave between building constraints restricting flexibility in the path. Additionally, the feasibility of the scenario is limited, as a realistic wind environment in the urban setting would consist of head and tail winds through building corridors and possible wind shear at street intersections. Future work for this scenario consists of the implementation of dynamic wind fields while also providing multiple CSCs defined by the turning radius of the vehicle within the Triplanner algorithm. Providing different CSCs from the initial point to the terminal point would allow for more flexibility in the chosen path and increase the opportunity for convergence in extreme wind conditions.

5.6 Contingency Planning

There may be times within a flight mission when dynamic constraints or mission priorities require the SUAS to change a flight plan mid-way through a defined scenario. This may be the result of an unpredictable movement of a population center or convoy that is undesirable to fly over, or intelligence has changed the initial targeting information. As the vehicle proceeds through a set of simplexes, it would now be required to generate and re-route to a newly defined CSC. In reference to the two-dimensional problem scenarios and in the extreme case, the aircraft is restricted between two building constraints in the urban environment with a minimum width of two times the minimum turning radius as defined by the Triplanner algorithm. Given this minimum constraint, a set of three minimum radius turns can be accomplished to reverse the SUAS direction within the current CSC. This set of maneuvers are shown in Figure 51. Here, the solid red lines define the minimum width of a CSC, δ defines

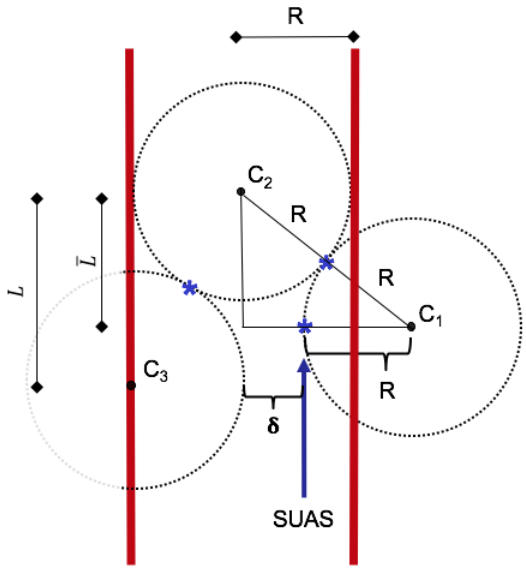

Figure 51. Contingency Algorithm Development

the off-set distance of the current SUAS position to the center of the CSC, and R the

minimum turning radius based on the vehicles minimum speed. The vertical start distance of the vehicle can then be defined with these two parameters as

$$\bar{L} = \sqrt{4R^2 - (\delta + R)^2} \tag{5.38}$$

where L is defined as a unique case when the SUAS is in the center of the CSC, requiring $\delta = 0$. The minimum look-ahead distance required to make the maneuver is simply $\bar{L} + R$.

In order to minimize the required area to perform the maneuver while accomplishing the task in minimum time, a set of three minimum radius turns can be determined in closed-form solution, producing the radial distance and time required for each segment of the maneuver.

Contingency Operation Algorithm Development.

The contingency algorithm starts with the vehicle parallel to the left CSC boundary defined by position (x_0, y_0). The center of the circles are comprised of the parameters shown in Figure 51 and defined as

$$C_1 = (x_0 + R, y_0) \tag{5.39}$$

$$C_2 = (x_0 - \delta, y_0 + \bar{L}) \tag{5.40}$$

$$C_3 = (x_0 - (\delta + R), y_0 - (L - \bar{L})). \tag{5.41}$$

Each of the three circular paths can be described with

$$(x - C_x)^2 + (y - C_y)^2 + R^2 = 0. \tag{5.42}$$

Equating each of the two connecting circles, the coordinate position of the tangent points can be determined given the input of C_i and R. Knowing the initial point of the aircraft, (x_0, y_0), the distance traveled on each of the circular paths can be determined by

$$S = R(\theta_2 - \theta_1) \tag{5.43}$$

for θ is equal to the angular distance between consecutive tangent points, beginning with the start location (x_0, y_0). The algorithm requires the final position of the aircraft to be parallel to the initial starting vector, which results in a quarter turn on the final circular path, given this scenario development. With the minimum velocity of the aircraft, v, the flight time spent on each arc is determined by

$$t = \frac{S}{v}. \tag{5.44}$$

With the equation and parameters presented, a closed-form solution for the contingency maneuver can be attained.

Contingency Operation Results.

The analytical solution provides a path that contains the tangent point coordinates and the time spent on each arc. The position coordinates and heading angles in-between each tangent point can be interpolated on each circular region. This generic path solution is shown in Figure 52 and illustrates the solution with a small off-set distance, δ to the CSC center line. Like Triplanner, this solution is a geometric path solution which does not consider rate limits on the state and control parameters. Therefore, this solution is used to seed the NLP of the optimal control solver. Given the optimal control problem defined in in Section 5.2, Figure 53 shows the contingency

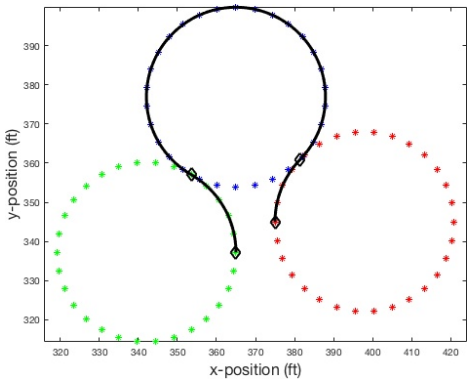

Figure 52. Contingency Algorithm Analytical Results

path solution. In this scenario, the black asterisks indicates the original flight path

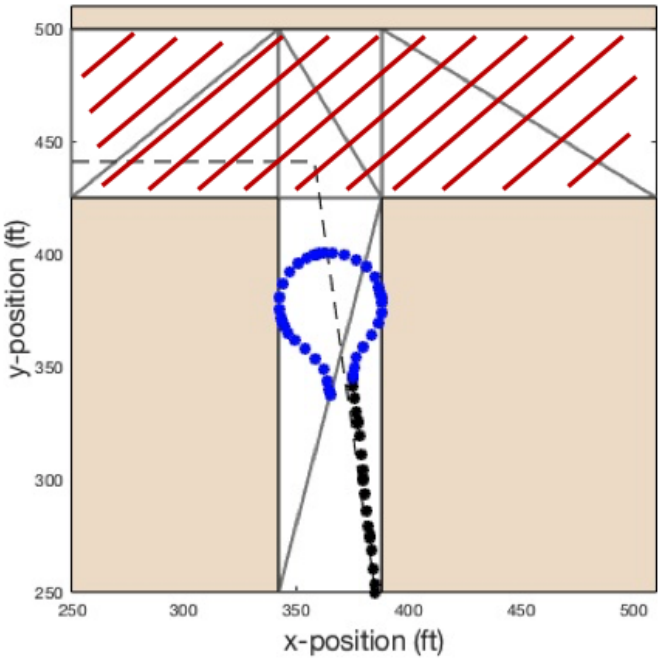

Figure 53. Contingency Algorithm Optimal Path Solution

of the SUAS. The transition to the blue asterisks shows the path of the contingency operation. The implementation of this solution requires the minimum look-ahead distance of $\bar{L} + R$, plus the required time to calculate the maneuver. The solution for the contingency operation was accomplished in 0.96 seconds with an objective time

of 15.6 seconds. The objective time is significant as this is the time available for the aircraft to receive additional mission information and calculate the subsequent CSC and flight path. The control parameters for the maneuver are shown in Figure 54. Here, the contingency operation can be seen to begin at the 3.5 second mark with

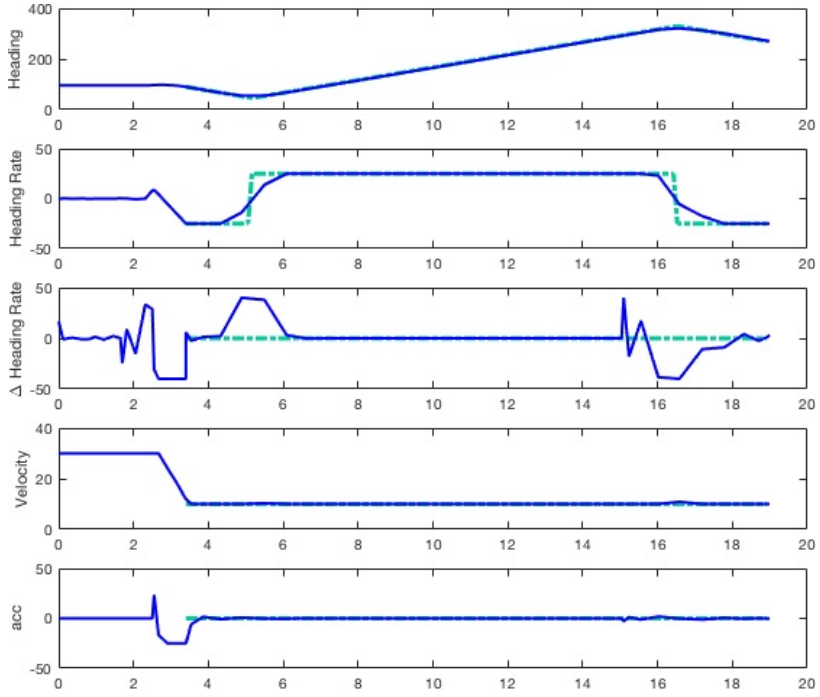

Figure 54. Contingency Algorithm Optimal Control Demonstration

the analytical path solution shown in green. The first plot shows the heading as the vehicle traverses around each of the three arcs. The second plot shows the heading rate and illustrates the implementation of the rate limits shown in third plot. Finally, the last two plots show the velocity and acceleration remaining constant throughout the operation so that a minimum radius turn can be achieved.

Contingency Operation Conclusions.

The simplex methodology for optimal path planning in highly constrained environments has proven to be a valid, efficient, and accurate method for determining

optimal path solutions. The scenario outlined in this section provides the ability to re-route the aircraft mid-way through a two-dimensional mission, should the current CSC become obstructed. The contingency operation is guaranteed from Triplanner, as the CSC is constructed to have a minimum width of two times the minimum turn radius of the vehicle. As long as the minimum path distance to the obstruction is provided, a contingency operation can be performed.

5.7 Summary

This section provided four variations to the two-dimensional CSC optimal control problem. These variations covered multiple waypoint analysis, implementation of soft keep-out zones, incorporation of wind parameters, and contingency operations should a CSC become closed off. These solutions provide standard scenarios and a breadth of capability that can be accomplished within the CSC construct and further illustrates the capability of the solution methodology. These results provide an optimistic view as more complex mission scenarios and different cost functions are implemented, fast and accurate solutions can be attained.

VI. Conclusions and Recommendations

6.1 Conclusions

This chapter presents the conclusions, contributions to the field of aeronautical engineering optimization theory for path planning, and recognizes potential areas for future research. The objective of this work was to define a feasible flight path through a constrained environment, quickly and efficiently, with a hybrid method combining optimal control direct orthogonal collocation methods with fast geometric path planning techniques. This objective was successfully accomplished by answering the following four questions:

1. **How do you formulate the two-dimensional optimal control problem for optimal trajectories in complex urban environments?** The work herein evaluated the formulation of constraint functions for modeling a realistic constrained environment. An analysis was conducted to compare simple circular and elliptical shapes, superquadrics, and polygonal functions. Each of these methods resulted in an increased burden on the NLP as more constraints are applied to the field, thus increasing computation time and reducing the probability of convergence. Simplex techniques were evaluated to remove the constraints from the domain and restrict the solution space to a set of connecting simplexes. This work formulates the minimum time optimal control problem with a phased approach in the barycentric coordinate frame, resulting in the optimal path solution through a defined CSC.

2. **Can the computational speed and robustness of convergence to the two-dimensional optimal control problem be improved by formulating the problem within a CSC construct using fast geometric path**

planning techniques? This work evaluated the two-dimensional optimal control problem with four different scenarios. First, a simple constraint field was implemented and results were shown for six different methods of solving the constrained optimal control problem. Results indicated the most computationally efficient algorithm formulated the problem within a CSC structure while using fast geometric path planning tools to seed the NLP. Convergence rates were increased by transitioning to the barycentric coordinate frame, thus eliminating problem specific parameter bounds. The remaining three scenarios increased the constraint field to 37 polygonal constraints resembling building structures. These scenarios demonstrated the CSC methodology, provided multiple waypoint solutions, and minimized incursion to keep-out regions. Evaluating the multiple waypoint scenario, discontinuities in the planning algorithm were addressed in the optimal control problem and moving the vehicle control from velocity to acceleration reduced the vehicles turning radius and allowed the aircraft to reach waypoints in tightly constraints regions of the Chicago map. In the keep-out region scenario, sigmoid functions were incorporated to allow for minimal incursions to unavoidable constraints, allowing for flexibility in designing constraint maps for vehicle path planning. Results showed the implication of including minimum flight regions within the CSC construct where the regions can only be completely avoided if it lies entirely within the CSC boundaries. By implementing the CSC method with these scenarios, solutions were attained to problems that previously could not be evaluated with optimal control solvers.

3. **How do dynamic constraints and constant wind fields affect the flight path solution acquired from a simplex discretization?** The work herein demonstrated simplex discretization techniques decrease computation time while providing accurate solutions to the two-dimensional optimal control problem for

path planning. However, since the solution is restricted to the defined CSC, dynamic constraints or mission requirements may result in that CSC becoming obstructed or closed off. Therefore, a contingency flight path was developed allowing for a turn around maneuver to be completed within the CSC. During this maneuver, time is available to generate a new CSC and flight path based on the updated mission requirements. Additionally, a constant wind field was successfully incorporated into the CSC construct of the control problem to demonstrate the path planning capability based on a priori knowledge of the wind fields.

4. **How can the three-dimensional problem be formulated using simplex discretization techniques?** The final scenario evaluated in this work demonstrated the extension of the two-dimensions problem to the third dimension. Path planning through a simple constrained environment was evaluated using superquadric constraint functions as well as tetrahedron discretization techniques to eliminate the constraints from the search field. The optimal control problem for the CSC method was formulated with control on the SUAS angle rates and velocity. With a simplistic constraint field, this formulation was more than adequate. However, as more realistic constraints are added to the scenario, extending the control to the vehicle's acceleration should be implemented in order reduce the turn angles and allow for navigation in tight corridor regions. Comparing the CSC method with superquadratic constraints, the CSC method once again proved to be superior in computation time and accuracy. However, extending the problem to three dimensions requires a CSC comprised of tetrahedrons. This discretization was accomplished in this work for simple polygonal constraint shapes as a proof-of-concept, however as more complex constraints are implemented into the field, this discretization may become problematic.

Collectively, methodologies were developed to provide fast, accurate solutions to the constrained optimal control problem, closing the gap to real-time, onboard path planning operations. Solutions can now be attained regardless of the number of constraints in the domain and independent of problem specific parameter bounds within the optimal control solver. The Triplanner algorithm provided both the feasibility of a path solution as well as a path solution itself, allowing for optimal solutions to be generated quickly and accurately. The work presented provided a wide range of scenarios, demonstrating the robustness of the methodology for solving optimal control problems for SUAS path planning.

6.2 Contributions

This research provided the following contributions to the field of optimal control path planning for SUAS:

1. Referring to Figure 2 in Chapter I, algorithms were developed and implemented to accomplish the work herein. A geometric path planner was developed for the three-dimensional scenario, defining a tetrahedral discretized space, a CSC, and in initial path solution. All data conditioning algorithms were developed to formulate the CSC connectivity matrix and generate a properly formatted initial guess. Within the optimal control solver GPOPS-II, algorithms were developed to translate the problem to barycentric coordinates and to build appropriate data structures within a phased approach through the defined CSC. These algorithms were essential to methodology and developed for solutions to a wide range of constrained optimal control problems.

2. Constraint modeling was shown to be too computationally expensive for real-time optimal control problems through highly constrained fields. Therefore

defined methods demonstrated solving highly constrained optimal control problems for SUAS path planning by eliminating all hard keep-out constraints from the NLP's domain and reducing problem specific parameter bounds to aircraft specifications.

3. Demonstrated the methodology for the inclusion of multiple waypoints, minimizing incursions to keep-out regions, incorporating constant wind fields and provided contingency path solutions within the CSC.

4. Provided a foundation for SUAS optimal path solutions in three-dimensions by performing a tetrahedron discretization of the search domain, acquiring a feasible CSC, and implementing the simplex methodology into the optimal control solver.

6.3 Potential Future Research

1. A multitude of different scenarios could be developed with varying cost functions to accomplish different mission objectives. This could include the requirement that an aircraft arrives at a desired location within a CSC at a specified time. Aircraft deconfliction could be considered to assure that two aircraft do not occupy the same simplex of the CSC at the same time. Or finally, consider multiple aircraft scenarios upon which all vehicles must converge onto the same target at the same specified time.

2. Each scenario solved in this work minimized the time of flight from the start position to the terminal point. In the urban environment, this may involve several max rate turns which can diminish the battery life of the SUAS and reduce the mission flight time. It would be beneficial to examine scenarios upon which the heading rate and throttle control are minimized, thus maximizing the

battery life of the vehicle. To accomplish this, different CSCs would need to be evaluated within the optimal control problem. Within the two-dimensional approach, Triplanner yields results based on the minimum turning rate of the vehicle. This turn rate can be varied over the range of the vehicles velocity to acquire different path solutions. Figure 55 illustrates the different path solution based on the vehicles speed.

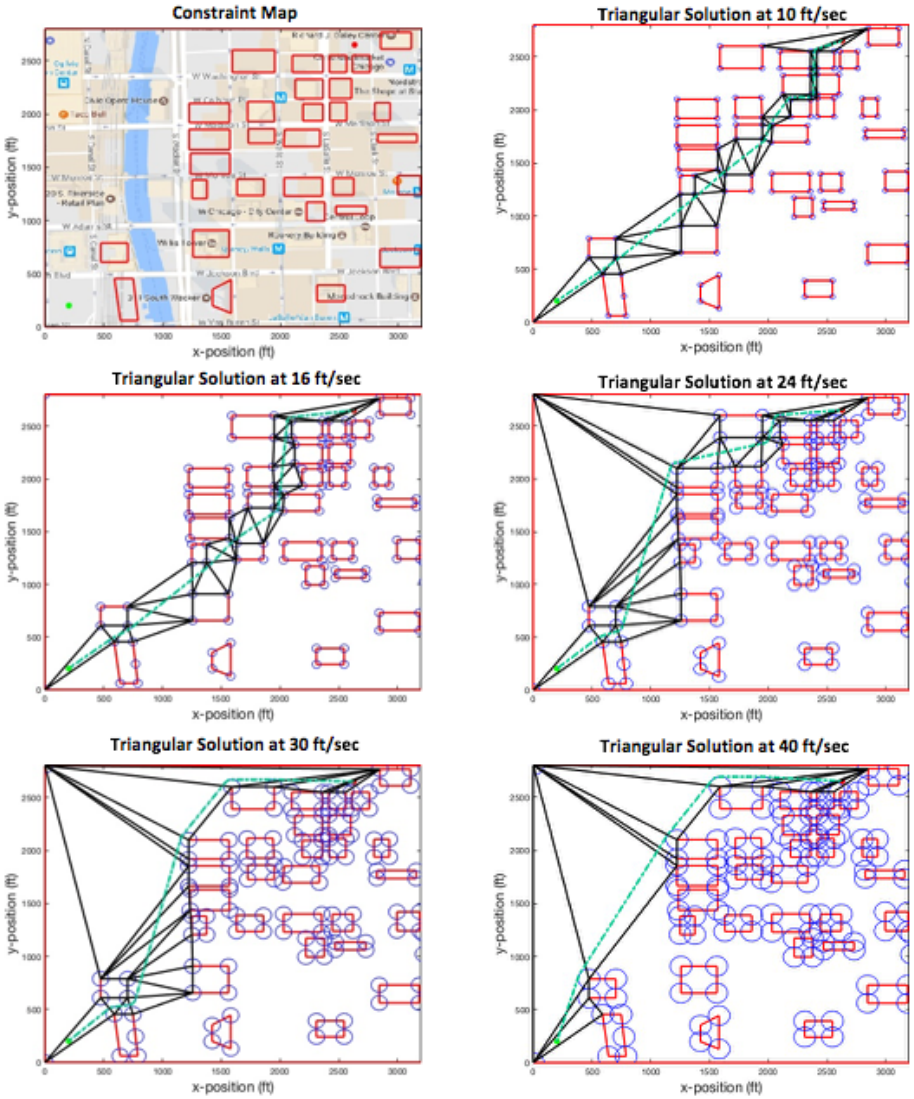

Figure 55. Triplanner Multiple Path Solutions. [Map Data @2017 Google]

Six figures are shown starting with the constraint map. Each subsequent figure increases the speed of the vehicle through the values of $10ft/s$, $16ft/s$, $24ft/s$, $30ft/s$, and $40ft/s$. Each speed produces a different path solution that can be evaluated based on the desired cost function within the optimal control solver.

3. This work uses the 2010 version of the Triplanner toolkit. This package is a fast geometric path planner that does not consider heading requirements at the initial and terminal point. A 2014 version of Triplanner is used under contract with AFRL and is recommended for future work, although an open-source, government owned version is the preferred end state. The 2014 version of the toolkit allows for initial and final heading constraints and is more robust to adding and removing large number of constraints quickly within the software package.

4. For scenarios involving multiple waypoints, exclusion to keep-out regions, or constant wind fields, computation times may be excessive. To reduce these computation times, an analysis should be completed to format the optimal control problem within the construct of a finite horizon or modern predictive control, evaluating the best approach to solve portions of the larger problem. Within the simplex discretization method, these smaller optimal control problems should solve very quickly in a consecutive manner.

5. The SUAS parameters used in this body of work were not tied to a specific aircraft, rather generic parameters were used to show the feasibility of the concept. The logical next step is to define an aircraft for flight testing and tune the algorithm to the characteristics of the specified vehicle. Preparing for a flight test, one should consider safety of flight issues with the constraints. Since the path planning algorithm does not include any safety buffer between the aircraft and

the building constraints, this safety buffer needs to be incorporated into the constraint design. Flight testing can be accomplishment with the two-dimensional algorithm or the three-dimensional algorithm.

For the two-dimensional test, two realistic options exist. The first is to fly a waypoint path defined with the output of Triplanner. This requires waypoints to be extracted from the Triplanner algorithm and spaced appropriately as Triplanner only gives the the first and last point on the straight sections of the Dubins path and points defined by a degree off-set on the minimum radius turns. This path would be a Dubins path, but would not be the optimal path within the CSC. It would however, give a realistic solution in a highly constrained environment. The second option would be to fly the heading rate vector and velocity vector output from the optimal control solver. Flying the control vectors lends to a more realistic implementation of flight testing the optimal flight path.

The three-dimensional flight test would need to be simplified significantly as the current algorithm can only handle rectangular prism constraints. Additionally, unlike in Triplanner, there are no guarantees for feasibility in the CSC. This must be considered in choosing the domain space. Further, the seed to the NLP for the three-dimensional algorithm is a midpoint solution between each consecutive simplex. This provides a very poor flight path and would not be recommended for a waypoint flight test. Therefore, it is recommended to fly the control vectors from the output of the optimal solver or waypoints acquired from the optimal solution.

6. Following a successful flight test, the code for the algorithm should be combined for seamless transitions between the software packages. It is recommended to run the algorithm on a Linux based operating system. There are four packages that need to be re-coded and seamlessly combined. The first is a script to pick

points off a constraint map to load the constrained field. This script should also contain all vehicle specific parameters and the start and final destination. The first script needs to call the geometric path planner, which would require minimal alterations to the code used herein. Third, a script needs to incorporate the output of the geometric path planner and format the data for the optimal control problem. This code has been written in MATLAB® and would just need transferred to a more suitable language. Finally, it is desired to call the NLP solver directly. Therefore, a script is required to build the parameter bounds, format the initial guess, and set up the input parameters for the NLP solver, to include the gradient and hessian of the dynamic functions. Once completed, this code will further close the gap to an operational, real-time system for optimal path planning.

7. Extending the algorithm to three-dimensional space could be accomplished with two different algorithms. The first would extend the concepts defined in this body of work, flying the aircraft through a tetrahedral CSC. This process would require increased fidelity in discretizing the space while providing guarantees to flight feasibility similar to the funnel algorithm in the two-dimensional approach. The second option for the algorithm is to consider a multi-level, two-dimensional flight path. Understanding and implementing a procedure to mesh the two-dimensional altitude levels will be challenging, but may be a more realistic approach for implementing a three-dimensional solution in the near-term.

6.4 Summary

The body of work herein has shown a successful methodology for acquiring fast and accurate solutions to the constrained optimal control path planning problem. By formulating the problem within a simplex mesh, parameter bounds were simplified

and the optimal control problem was solved in a phased approach with direct orthogonal collocated methods. Geometric path planning algorithms proved to be an efficient and accurate method for seeding the NLP, further decreasing computational time. Although this work does not provide a beginning to end solution for MUM-T operations, it does provide a foundation for a fast, accurate, and reliable solutions. Collectively, these algorithms begin to close the gap for real-time onboard SUAS path planning. Further, this work provides a methodology for MUM-T operations, where a SUAS can operate with minimal command from host aircraft and successfully accomplish the desired mission.

Appendix A. Presentations

Appendices B through G contain a conference or journal paper published as part of this work. Additionally, many presentations were given at local conferences or within the AFIT and local community. A list of all presentations under this research are included here.

1. **ASME Dayton Engineering Science Symposium**, November 2016. *"Passive Techniques for Target Localization using Trajectory Optimization"*

2. **AIAA Sponsored Lunch and Learn**, December 2016. *"SUAS Optimal Control using Triangulated Mesh"*

3. **AIAA Dayton & Cincinnati Aerospace Science Symposium**, March 2017. *"Simplex Methods for Optimal Control of Unmanned Aircraft Flight Trajectories"*

4. **AFIT Controls & Optimization Brown Bag**, August 2017. *"Simplex Control Methods for Convergence of Small Unmanned Aircraft Flight Trajectories"*

5. **ASME Dynamic Systems Controls Conference**, October 2017. *"Simplex Methods for Optimal Control of Unmanned Aircraft Flight Trajectories"*

6. **ASME Dayton Engineering Science Symposium**, October 2017. *"Optimal Path Planning for SUAS Waypoint Following in Urban Environments"*

7. **AIAA SciTech Information Systems Conference**, January 2018. *"Simplex Optimal Control Methods for Urban Environment Path Planning"*

8. **AIAA Dayton & Cincinnati Aerospace Science Symposium**, February 2018. *"Optimal Path Planning for SUAS Target Observation through Constrained Urban Environments using Simplex Methods"*

9. **IEEE Big Sky Aerospace Conference**, March 2018. *"Optimal Path Planning for SUAS Waypoint Following in Urban Environments"*

10. **AFRL/RW Controls & Optimization Outreach**, March 2018. *"USAF Optimal Control Applications"*

11. **AIAA American Controls Conference**, June 2018. *"Optimal Path Planning for SUAS Target Observation through Constrained Urban Environments using Simplex Methods"*

Appendix B. Dynamic Systems Controls Conference 2017

Appendix B contains the first paper published in this research effort. This work contains the constraint analysis and feasibility approach to the optimal control path planning problem, providing the foundation for the core research. It was published and presented at the ASME Dynamic Systems Controls Conference in Tysons Corner, Virginia, October 2017.

Proceedings of the ASME 2017 Dynamics Systems and Control Conference
DSCC 2017
October 11-13, 2017, Tysons, VA, USA

DSCC2017-5031

SIMPLEX METHODS FOR OPTIMAL CONTROL OF UNMANNED AIRCRAFT FLIGHT TRAJECTORIES

Michael D. Zollars
Department of Aeronautics & Astronautics
Air Force Institute of Technology *
Wright-Patterson AFB, OH 45433
Email: michael.zollars@afit.edu

Richard G. Cobb
Department of Aeronautics & Astronautics
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433

ABSTRACT

The feasibility of using a constrained Delaunay triangulation method for determining optimal flight trajectories of unmanned air vehicles in a constrained environment is explored. Current methods for developing optimal flight trajectories have yet to achieve computational times that allow for real-time implementation. The proposed method alleviates the dependency of problem specific parameters while eliminating constraints on the Non-Linear Program. Given an input of obstacles with n vertices, a constrained Delaunay triangulation is performed on the space. Converting the vertices of the triangulation to barycentric coordinates on a phased approach defines the state bounds and max time for each phase. With two-dimensional aircraft dynamics, direct orthogonal collocation methods are performed to compute the optimal flight trajectory. Results illustrate computational times and feasibility of Small Unmanned Aircraft System flight trajectories through polygon constraints.

INTRODUCTION

In the discipline of optimization and control, path planning techniques have been used over a wide range of applications to find optimal flight trajectories through constrained environments. These path constraints may consist of terrain, infrastructure, or no-fly zones that must be avoided for the safety of the aircraft or the concealment of the air mission. Often, Small Unmanned Aerial Systems (SUAS) are teamed with manned aircraft to minimize the dangers the aircrew may encounter due to these constraints [1]. In the case of target localization, a host aircraft can remain at a safe altitude while a SUAS is deployed into a threat region to acquire and verify target coordinates. Once the initial target coordinates have been verified, the SUAS is required to determine a feasible path to a second target region. The autonomous localization of the target has been studied in [2] and this paper focuses on the autonomous path development from the initial target region to the second target region through a constrained field.

In previous work studying constrained optimal flight trajectories [3–7], significant issues have been illustrated that prevent the algorithm from reaching computation speeds required for onboard operations. For numerical solutions, an initial guess must be provided to the Non-Linear Program (NLP) solver and problem specific parameters must be estimated before a solution can be determined. The accuracy of these inputs will affect both the computational speed and the convergence to an optimal solution. This paper focuses on increasing computational speed and guaranteeing convergence by eliminating constraints from the search space through triangulation and performing a coordinate transformation to standardize the input parameters to the NLP solver.

OPTIMAL CONTROL

Due to the complexity of most optimal control problems, analytical solutions cannot be obtained and therefore numerical methods are employed to reach feasible solutions. Two common numerical approaches are described as the indirect method

and the direct method. Indirect methods apply the calculus of variation to transform the optimal control problem into a Hamiltonian Boundary-Value Problem (HBVP). This method produces a highly accurate solution and assures the first-order optimality conditions are solved. However some disadvantages include a small radii of convergence and the requirement to analytically derive the HBVP. A good guess for the states, control and co-states is required for convergence, which often becomes time consuming and problematic, specifically for the co-states which have no obvious physical meaning to the problem [8].

To avoid these issues, direct methods transcribe the infinite-dimensional optimal control problem into a finite optimal control problem with algebraic constraints, otherwise known as an NLP [9]. There have been many methods developed to transcribe the optimal control problem into an NLP, including direct shooting methods [10], state and control parameterization methods [11], and direct orthogonal collocation methods [3, 12]. The focus of this research will be on direct orthogonal collocation methods, also referred to as psuedospectral methods in the Aerospace field of study [9].

When solving an optimal control problem with direct orthogonal collocation, the continuous time optimal control problem is transcribed to a discretized nonlinear programming problem. This is accomplished with three main concepts; orthogonal collocation, polynomial approximation, and Guassian quadrature [13]. For this research, the continuous functions of the optimal control problem are approximated with a finite dimensional Lagrange polynomial basis [14]. The state x is approximated at a set of collocation points described as

$$\tilde{x}(\tau) \approx \tilde{x}_N(\tau) = \sum_{i+1}^{n+1} x_i L_i(\tau) \tag{1}$$

where x_i represents the weight function, $L_i(\tau)$ is the Lagrange polynomial basis

$$L_i(\tau) = \prod_{\substack{i=0 \\ j \neq i}}^{n} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{2}$$

and τ represents an affine transformation of the time t on the interval from $(-1,1)$.With the problem discretized, Gaussian quadrature is used for differentiation or integration of the state and control. The error produced by this method can be greatly reduced by choosing the collocation points appropriately with Legendre or Chebyshev point placement to minimize the affects of Runge phenomenon.

CONSTRAINT MODELS

Many techniques have been employed to appropriately model path constraints in two-dimensional space. One of the most commonly employed techniques is to simulate a vehicle no-fly zone with circular or spherical constraint function [5, 12, 16] of the form

$$\|x_k(t) - \tilde{x}_n\|_2^2 > r_n. \tag{3}$$

Here, the difference between the position states of the k^{th} vehicle, x_k and the center of the no-fly zone, $\tilde{x}$ must be greater than a defined radius, r_n. These functions are differentiable and smooth and therefore can be handled with relative ease with an NLP solver such as *IPOPT* or *SNOPT*. The drawback is that the constraint is restricted to circular or spherical shapes. This may be adequate to represent a no-fly zone, however error will be introduced for polygon shapes representing city infrastructure or landscape that require square or rectangular shaped constraint models.

To minimize this error, superquadrics, have been used extensively in computer vision, computer graphics, and robotics [17]. The superellipse is a type of Leme curve which is defined as

$$\left(\frac{x_c}{a}\right)^m + \left(\frac{y_c}{b}\right)^m > 1 \tag{4}$$

where the semi-major and semi-minor axes are defined by a and b, x_c and y_c represent the center point of the object, and m is a rational number defining the curvature of the object. When solving problems with direct orthogonal collocation, the superquadric has been used to include constraint shapes other than simple circles or ellipsoids. Hurni, Lewis and Mohan used the superquadric in a constraint rich environment and found success in convergence but at the cost of computation time [18–20].

Additionally, polygon shaped constraints can be evaluated with ray-casting to determine if the collocation point is inside the constraint or in a feasible region in the search space [21, 22]. The polygons are described by a vector connecting each vertex to form a closed region. Unlike the superquadric, any polygon shape can be modeled with this method. The ray-casting algorithm returns both a Boolean variable to obtain a smooth constraint and the distance to the nearest polygon edge. This method is computationally expensive and can quickly become unsolvable in optimal control software as more constraints are added to the optimal control problem. Further, the derivative of the constraint function becomes steep at the constraint boundaries which can result in undesirable and non-optimal path solutions. The robustness of the algorithm to consistently converge to a solution is problematic as well, and computation times required for real-time operations are not feasible with this method.

Each of these methods are based on the premise that the constraint is contained inside the search space and the NLP solver must evaluate constraint equations to assure a feasible solution can be attained. The following section leverages research from the field of computer animation to eliminate the constraints from the search space by performing a triangular discretization dependent on the constraint field rather than the size of the search space.

COMPUTER ANIMATION

Direct orthogonal collocation methods discussed previously are capable of meeting the computational requirements if an adequate starting point of the states are given to the NLP solver. When approaching the constant altitude SUAS problem, comparisons can be made to path development of autonomous agents in the interactive virtual world. Virtual worlds are populated with autonomous virtual humans, or agents, where computed paths must take into account path length, time, and energy expended traversing the path [23].

The demand on the complexity of the virtual world combined with improved graphic capability has encouraged new techniques for achieving intelligent navigation for the next generation virtual agent simulation [24]. Given the speeds of which these video games are played, path planning algorithms must perform efficiently under limited time budgets to traverse the autonomous agent from one end of a building to the other. Often greedy algorithms are chosen in which the first feasible path that is found is continuously improved according to the characterestic dynamics and a pre-determined threshold on computational time. The path available at the time required is then implemented [23, 25]. This can often result in a sub-optimal path, but in the virtual world of the gaming industry, the timeliness of the solution is more heavily favored over the most optimal route.

CONSTRAINED DELAUNAY TRIANGULATION

A Constrained Delaunay Triangulation(CDT) is a refinement of the Delaunay triangulation that forces a required segment as an edge of the triangulation. The formal definition can be found in [26] and is described as follows:

Definition. Let G be a straight-line constrained edge. A triangulation T is a CDT of G if each edge of G is also an edge of T and for each remaining edge e of T there exists a circle C with the following properties:

1. The endpoints of edge e are on the boundary of C.
2. If any vertex v of G is in the interior of C then it cannot be "seen" from at least one of the endpoints of e (i.e., if you draw the line segments from v to each endpoint of e then at least one of the line segments crosses an edge of G).

The CDT is made of constrained edges and unconstrained edges. When determining an optimal path, the solution can cross an unconstrained edge, but must avoid all constrained edges. Since the constrained edges are required, often formulation of the CDT contains a vertex or an edge that does not satisfy the Delaunay triangulation conditions, and therefore a CDT may not be equivalent to a Delaunay triangulation. An example of the differences can be seen below in Figure 1.

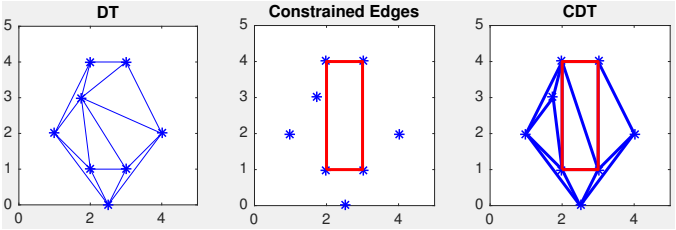

FIGURE 1. A: Unconstrained Delaunay Triangulation; B: Constrained Edge Segments; C: Constrained Delaunay Triangulation [26]

Here, Figure 1A shows a Delaunay triangulation without any constrained edges. Figure 1B implements a constrained edge into the field and Figure 1C illustrates the CDT with the constrained edges included in the triangulation.

Using this technique, constraints can now be forced into the discretization of the space. In computer animation, these constraints represent walls in buildings, tables, chairs, and other common structures autonomous agents are required to navigate around. Applying these constraints to the SUAS path planning problem, these boundaries illustrate hard constraint no-fly zones that are represented with polygonal shapes such as buildings, terrain, and restricted airspace.

TRIANGULATION-BASED PATH DEVELOPMENT

Path planning algorithms have been designed to determine the best route from an initial starting point to a goal location while satisfying conditions on constraints and dynamics. In the field of optimal control, the dynamics of the agent must be satisfied while obeying all constraints placed on the states, control, and path. In computer animation, path planning focuses on determining the best route for the autonomous agent while avoiding walls, furniture and other obstacles. Many algorithms have been developed to accomplish autonomous agent path planning with an emphasis on computational speed. Path solutions must be found quickly to give the appearance the agent is in fact making real-time intelligent decisions as the video game progresses [27].

A* Search Algorithm

The A* search algorithm gets its roots from Dynamic Programming and is an extension to Dijkstra's algorithm. The entire domain is searched with Dijkstra's algorithm, where as an A* search is guided by a heuristic. It is considered an "informed search" or "best-first search" as the algorithm searches the space until a solution is found. This solution may be sub-optimal and therefore the algorithm will continue to search the space, updating the solution if a lower performance index is found. Often a time limit is imposed on the algorithm and sub-optimal paths are accepted in order to find a solution in the quickest possible manner. In computer animation, this is a trade-off in order to achieve the computational times required for computer animation.

The initial node for the A* algorithm is given a cost value of $f(1) = 0$ and additional nodes, n, are searched in a tree like fashion. A heuristic is introduced and added to the cost function to give a priority in the direction of the search. Often the Euclidean distance is used as the heuristic and the overall cost function is

$$f(n) = g(n) + h(n) \tag{5}$$

where $g(n)$ is the cost evaluated at the current node and $h(n)$ is a heuristic that estimates the cheapest path from the current node to the endpoint [28]. At each iteration, the A* algorithm must determine which nodes to expand by selecting the path that minimizes the current cost, $g(n)$ and the cost to go, $h(n)$. In the case where $h(n)$ is set equal to 0, the result will be consistent with Dijkstra's algorithm [29]. The A* search algorithm is demonstrated herein through the search results of the Tripath toolkit.

Tripath Toolkit

Marcello Kallmann has provided an extensive review of path development with clearances in [30] and has toolkit algorithms designed to determine the shortest path while accounting for minimum clearance distances to all constraints [31]. Specifically, he focused on determining paths of shortest distances, efficiency of computation time, and maintaining proper clearance from obstacles along the calculated path. These algorithms have been implemented in the Tripath toolkit[1]. An overview of the development of the Tripath algorithm is discussed below. A more extensive review of the algorithm can be found in [30, 32].

First, let S define a set of n segments that form all the constrained edges in the domain. The set of all endpoints of each segment then form the set P. A constrained Delaunay triangulation, T, is then formed such that all segments of S are also segments of T and the Delaunay criterion are upheld. Next, Kallmann performs a test to assure that a disk of radius r can traverse through any given region. This enables an efficient computation of paths with arbitrary clearances. If a local clearance test fails, a refinement of the mesh is performed by redistributing the triangulation or adding a vertex point to a straight line segment of the set S. This may result in a new set of constrained edges, as a straight line segment could be subdivided into multiple sections. Once the triangulation has passed the local clearance test, the final mesh is termed a "Local Clearance Triangulation (LCT)".

A path through the LCT from a starting point p to a finish point q is defined as "free" if it does not cross a constrained edge. A free path will cross several unconstrained edges resulting in a "channel" formed of all traversed triangles. Using an A* algorithm in a discretized space formed by the triangulation, a geometric solution for the channel is determined. The channel formed by the A* search is dependent on the chosen cost function and will result in a shortest path search defined by the cost metric. A simplistic cost metric may consist of using the centroid of each triangle or instead using the midpoint of each unconstrained segment to determine the best channel. Kallmann settled on a cost function which begins the search from the midpoint of each triangle, but enhanced the search through a reference point connecting the point in the previously traversed edge to the final point q. If q is not visible on the straight line path, the nearest vertex is used for the traversed point while accounting for the radial offset distance form each vertex point.

After the channel is determined, a funnel algorithm is used to find the shortest path within the channel. The funnel algorithm was developed by Lee and Preparata, and Chazelle [33, 34] as cited by [35]. The algorithm has been used to calculate the shortest path under multiple applications; including path finding for autonomous agents [36], querying visible points in large data sets to define shortest paths [37], shortest paths for tethered robots [38] and robots in extreme terrain [39].

Given a corridor with a starting position p and final point q, the funnel algorithm defines the entrance point of the simplex as the apex, a, and the starting point in the funnel. Since the corridor has already been defined using the A* algorithm, the two vertices connecting the next simplex can be defined as u and v, with the third vertex w. If the straight line solution from a to w is feasible, a straight line path is chosen from a to w as shown in Figure 2A. Maintaining a as the apex, the third vertex of the next simplex is evaluated for the straight line path solution from a to w'as shown in Figure 2B. In the case where the straight line solution is not feasible, the vertex closest to the next point in the path is chosen as the new apex, a' and the algorithm continues as shown in Figure 2C. Further detailed explanation of the funnel algorithm can be found in [35]. To account for the local clearance around obstacles, a circular constraint with radius r is imposed on each vertex as illustrated in Figure 2D [40].

The Tripath toolkit performs a locally optimal search and is capable of achieving path solutions on the order of milliseconds for environments with 60K+ segments [30]. Although there is no guarantee that the solution is the global optimal path, the simplex corridor found and searched is free of constraints allowing for

[1] http://graphics.ucmerced.edu/software/Tripath/

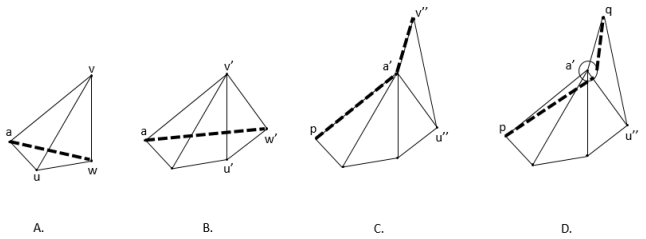

FIGURE 2. Implementation of the Funnel Algorithm on a Simplex Corridor

solutions consisting of straight line segments and max turn radius arcs.

Barycentric Coordinate Frame

Often when dealing with simplex shapes, the barycentric coordinate frame is preferred over traditional Cartesian coordinates. A barycentric coordinate system defines the location of a point within a simplex as a weighted measure to each of the vertex points, also referred to as areal coordinates in the context of triangles [41]. The barycentric coordinate system defines the side of the simplex as the axes, allowing for simple representations of lines, points, and perpendicular relationships [42].

To define the coordinate system, let q_1, q_2, ..., q_n be n vertices of a complex planar polygon G in $\mathbb{R}^2$ for $n \geq 3$. For the purposes of this research, when investigating problems in $\mathbb{R}^2$, triangulation techniques will be utilized and therefore $n = 3$. Any point, P, inside polygon G can be represented in barycentric coordinates defined with the vertices of G used as a basis as follows [43–45]:

$$\mathbf{P} = \sum_{j=1}^{n} \alpha_j \mathbf{p_j} \tag{6}$$

where α represents the weights, defined as a set of real coefficients whose added sum equals unity.

$$\sum_{j=1}^{n} \alpha_j = 1. \tag{7}$$

To ensure that each point remains inside the polygon, G,

$$\alpha_j > 0 \ \ \forall j \in [1...n]. \tag{8}$$

For the triangular relationship, $n = 3$, transformation from a barycentric coordinate frame to a Cartesian coordinate frame can be easily performed through a linear transformation of the coordinates represented as:

$$\mathbf{P} = \mathbf{QA} \tag{9}$$

where $\mathbf{P} \in \mathbb{R}^2$ defines the point location in the Cartesian coordinate frame, $\mathbf{Q} \in \mathbb{R}^{2\times n}$ represents the matrix of vertices in polygon G, and $\mathbf{A} \in \mathbb{R}^{\mathbf{n}}$ describes the weight matrix and the set of barycentric coordinates.

Coordinate Transformation

In order to simplify the optimal control solver input parameters, the problem is expressed in the barycentric coordinate system. Given a simplex shown in Figure 3, each vertex is defined in Cartesian coordinates as

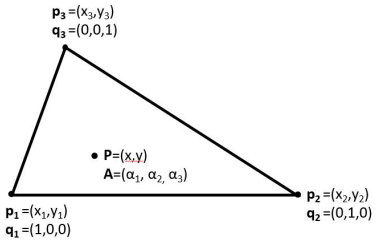

FIGURE 3. Barycentric Coordinate Frame

$$\mathbf{p_i} = (x_i, y_i) \ \ \forall i \in [1...n] \tag{10}$$

where n is equal to the number of sides of the simplex. The solution is limited to the constant altitude, two-dimensional plane and therefore $n = 3$. Each point within the simplex can be represented as an ordered triple of real numbers, representing the weighted distribution to each vertex. Each vertex is defined in barycentric coordinates as:

$$\mathbf{q_1} = (\mathbf{1}, \mathbf{0}, \mathbf{0}) \tag{11}$$

$$\mathbf{q_2} = (\mathbf{0}, \mathbf{1}, \mathbf{0}) \tag{12}$$

$$\mathbf{q_3} = (\mathbf{0}, \mathbf{0}, \mathbf{1}) \tag{13}$$

while any point within the simplex is represented with the corresponding weights to each vertex

$$\mathbf{A} = (\alpha_\mathbf{1}, \alpha_\mathbf{2}, \alpha_\mathbf{3}). \tag{14}$$

Given a set of barycentric weights, the Cartesian coordinates can be represented as

$$\mathbf{P} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \alpha_{\mathbf{i}} \mathbf{p_i}. \tag{15}$$

Expanding Equation 15 and representing the final weight α_3 in terms of the first two weights, the expression below represents the weights of the barycentric coordinate frame in terms of the Cartesian coordinates,

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = T^{-1}(\mathbf{P} - \mathbf{q_3}) \tag{16}$$

$$\alpha_3 = 1 - \alpha_1 - \alpha_2 \tag{17}$$

where T is a 2×2 matrix representing the vertices of the triangle as

$$T = \begin{pmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{pmatrix}, \tag{18}$$

and $\mathbf{P} - \mathbf{q_3}$ is a 2×1 vector summation of the Cartesian coordinates. Expanding Equation 16 gives the barycentric weights in terms of the vertex coordinates and the point location within the triangle.

$$\alpha_1 = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{det(T)} \tag{19}$$

$$\alpha_2 = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{det(T)} \tag{20}$$

$$\alpha_3 = 1 - \alpha_1 - \alpha_2 \tag{21}$$

Differentiating the weights with respect to the x and y position gives the following set of dynamic equations

$$\dot{\alpha}_1 = \frac{(y_2 - y_3)\dot{x} + (x_3 - x_2)\dot{y}}{det(T)} \tag{22}$$

$$\dot{\alpha}_1 = \frac{(y_3 - y_1)\dot{x} + (x_1 - x_3)\dot{y}}{det(T)} \tag{23}$$

$$\dot{\alpha}_3 = -\dot{\alpha}_1 - \dot{\alpha}_2. \tag{24}$$

SOLUTION METHODOLOGY

GPOPS-II is described as a computational tool for solving multiple-phase optimal control problems using variable-order Gaussian quadrature collocation methods [46]. Each phase is defined with a set of dynamic constraints, path constraints, integral constraints, and parameter constraints. Phases are linked through event constraints that relate information at the start and terminal point of each phase and allow for independent variables to be continuously transitioned through each phase [11, 47].

For this method, a solution through one simplex can be represented as one phase in GPOPS-II. Formulating the optimal control problem through a constrained Delaunay triangulation provides the basis for a trajectory solution that traverses through a corridor of simplexes (as described in the previous section) each represented as a single phase, linked together to determine the optimal solution. It is recognized here that the solution found by this method is dependent on the simplex corridor that is presented to the optimal solver. This corridor may not provide the global optimal solution, rather a locally optimal solution may be determined. Therefore, this paper focuses on determining a feasible solution consistently and efficiently as opposed to the necessity of arriving at the global optimal trajectory for every simulation. It will however provide an admissible solution satisfying the dynamic constraints.

The following sections define the optimal control problem in terms of a phased approach. Each simplex will be represented in barycentric coordinates with appropriate dynamic constraints, path constraints, and parameter constraints. The number of phases required to be solved is problem specific and defined after the space has been discretized into a CDT. The optimal control solver will search for a path through a simplex corridor that is defined prior to initiating the optimal control problem. The number of solution phases is represented with the variable P.

SUAS Dynamics

The dynamics of the aircraft are formulated with a three state model, representing the position of the SUAS in the $x(t)$, $y(t)$ directions and the heading angle, $\theta(t)$. The control for the aircraft is the heading angle rate, γ. The SUAS dynamics are described below as a relationship between the states and the controls

$$\dot{x}^{(p)} = (v)cos(\theta) \;\; \forall p \in [1...P] \tag{25}$$

$$\dot{y}^{(p)} = (v)sin(\theta) \;\; \forall p \in [1...P] \tag{26}$$

$$\dot{\theta}^{(p)} = \gamma \;\; \forall p \in [1...P] \tag{27}$$

where v represents the aircraft velocity which is held constant in this work.

Optimal Control Problem

By discretizing the search space with a CDT, a phased approach is taken in the optimal control problem. Each triangle is solved as a single phase in GPOPS-II and each phase is connected through event constraints. The dynamics constrain the path through each triangle in barycentric coordinates. Combining the described steps, the optimal control problem formulation has been consolidated as follows.

Minimize the cost functional

$$J = \int_{t_0^{(p)}}^{t_f^{(p)}} dt \ \ \forall p \in [1...P] \tag{28}$$

$$\tag{29}$$

subject to the dynamic constraints

$$\dot{\alpha}_1^{(p)} = \frac{(y_2-y_3)(v)cos\theta+(x_3-x_2)(v)sin\theta}{det(T)} \ \ \forall p \in [1...P] \tag{30}$$

$$\dot{\alpha}_1^{(p)} = \frac{(y_3-y_1)(v)cos\theta+(x_1-x_3)(v)sin\theta}{det(T)} \ \ \forall p \in [1...P] \tag{31}$$

$$\dot{\alpha}_3^{(p)} = -\dot{\alpha}_1^{(p)} - \dot{\alpha}_2^{(p)} \ \ \forall p \in [1...P] \tag{32}$$

$$\dot{\theta}^{(p)} = \gamma \ \ \forall p \in [1...P] \tag{33}$$

with the control

$$u^{(p)} = \gamma \tag{34}$$

and the state vector defined as

$$X = (\alpha_1, \alpha_2, \alpha_3, \theta) \tag{35}$$

with boundary conditions given as the initial and final constraints

$$X^{(1)}(t_0^{(1)}) = ((\alpha_1)_0, (\alpha_2)_0, (\alpha_3)_0, (\theta)_0) \tag{36}$$

$$X^{(P)}(t_f^{(P)}) = ((\alpha_1)_f, (\alpha_2)_f, (\alpha_3)_f, (\theta)_f) \tag{37}$$

and path constraints

$$0 \leq \alpha_1^{(p)} \leq 1 \tag{38}$$

$$0 \leq \alpha_2^{(p)} \leq 1 \tag{39}$$

$$0 \leq \alpha_3^{(p)} \leq 1. \tag{40}$$

SIMULATION

To verify the presented methodology, a two-dimensional three constraint optimal control problem was developed. For each scenario, the initial and final aircraft locations are defined as

$$(x_0, y_0, \theta_0) = (1, 0.5, free) \tag{41}$$

$$(x_f, y_f, \theta_f) = (9, 9.5, free). \tag{42}$$

Four constraints which represent areas the SUAS can not traverse, such as buildings, walls, terrain, or restricted air space are included. These constraints are modeled as polygons as

$$\begin{bmatrix} x \\ y \end{bmatrix}_1 = \begin{bmatrix} 0 & 10 & 10 & 0 \\ 0 & 0 & 10 & 10 \end{bmatrix} \tag{43}$$

$$\begin{bmatrix} x \\ y \end{bmatrix}_2 = \begin{bmatrix} 2.5 & 3 & 3 & 2.5 \\ 0 & 0 & 7 & 7 \end{bmatrix} \tag{44}$$

$$\begin{bmatrix} x \\ y \end{bmatrix}_3 = \begin{bmatrix} 4.25 & 5.25 & 5.25 & 4.25 \\ 4.5 & 4.5 & 5.5 & 5.5 \end{bmatrix} \tag{45}$$

$$\begin{bmatrix} x \\ y \end{bmatrix}_4 = \begin{bmatrix} 6.5 & 7 & 7 & 6.5 \\ 3 & 3 & 10 & 10 \end{bmatrix}. \tag{46}$$

Six simulations were evaluated in total. The first three were performed without triangulation and modeled the constraints with traditional methods of circles, ellipsoids, superquadrics, and polygon functions. The fourth method evaluates a technique that discretized the space through a CDT and optimizes a path through a corridor of defined simplexes. The initial guess for this method is the path connecting consecutive midpoints of each unconstrained edge in the search corridor. The fifth method examines the path solution from the Tripath toolkit which provides a solution based on a heuristic search algorithm. The final hybrid method combines the previous two, using the Tripath solution as an initial guess for the optimal control solution in GPOPS-II. Each model is compared with computation time, objective time, mesh tolerance, and initial guess requirements.

The first simulation models the constraints with simple shapes such as circles and ellipsoids where solutions can result in quick convergence to the optimal trajectory. However, the polygonal shape constraint requirement cannot be met and large errors will exist for a circular shape representing a polygon. Further, when solving with collocation methods such as GPOPS-II, collocation points will jump through a constraint dependent on the spacing of the collocation points. Figure 4A below illustrates a failed solution based on a poor guess. As the guess is improved in Figure 4B, an optimal solution is found, however the path violates the constraint as shown in Figure 4C.

In the second simulation, the superquadric can better represent polygonal shapes such as squares or rectangles by increasing the power, *m*, in Equation 4. Figure 5A below shows the same constraint field as the previous simulation, however now the polynomial order of the constraint function has been raised to 100. A feasible solution could only be found when the mesh tolerance was increased to 10^{-5} and a perfect initial guess was given to the NLP solver.

These constraint functions are more representative of polygonal shapes, however the flight path still cuts the corner of the constraint as the collocation points round the corner of each constrained edge as shown in Figure 5B. Further, the solution still

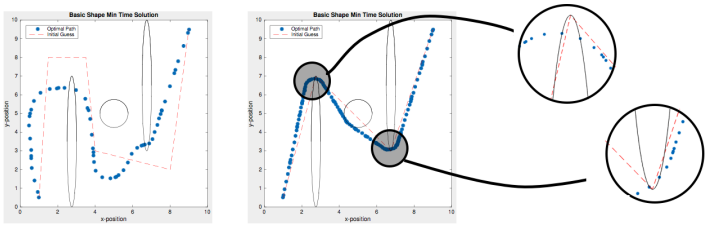

FIGURE 4. Simple shape constraint functions, illustrating converged solution for small semi-major axis

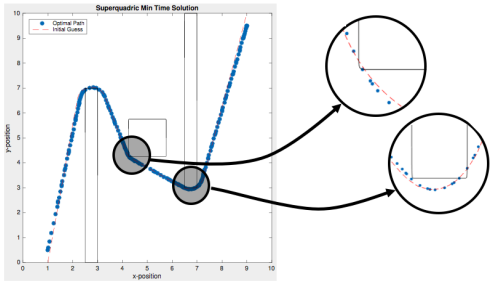

FIGURE 5. Superquadric constraint functions, illustrating converged solution for small semi-major axis with poor initial guess

requires a perfect guess and computation time remains between 10 and 20 seconds.

The third simulation utilizes point in polygon test function which allows for the modeling of any polygonal shape. Solutions presented in GPOPS-II are feasible, but the required computation time exceeded the limits for on-board operations. Additionally, as more constraints are added to the optimal control problem, the computational requirements become unyieldy.

Ultimately, the convergence of a solution is highly dependent on the user settings of GPOPS-II. The mesh tolerance can be increased to allow for tighter discretization of the space and to include more collocation points, however, this increases computation time and may result in convergence failure if the mesh is too tight for the given problem. Additionally, the initial guess plays a significant role in the solution search space. A poor initial guess may result in a local minimum solution rather than the global solution. Further, the better the guess the solver receives, the quicker the convergence to a solution. Finally, the shape of the constraints have an effect on the feasibility of the solution as well. Collocation points may cut the corner of the edge of a constraint or may even skip over the constraint if the collocation points are not spaced properly. Each of these factors are problem specific and require different dependencies on the GPOPS-II user settings. The next three simulations focus on eliminating these constraints from the problem formulation and minimizing the input requirements to achieve a feasible solution within computation times sufficient for on-board operations.

The fourth simulation triangulates the space using the constrained Delaunay triangulation function, *delaunay*, in Matlab®. A search corridor is selected with dynamic programming and an initial guess is determined by connecting the midpoint of each unconstrained edge of the search space. The solution is shown in Figure 6.

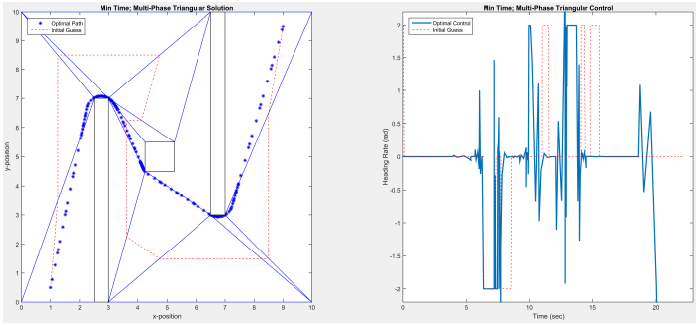

FIGURE 6. Constrained Delaunay Triangulation Solution

In the left image, it can be seen that optimal flight path determined through the simplex corridor does not violate any hard constraint. The right image illustrates the heading rate control over the flight time of the simulation. The control can be further refined to fully exhibit Pontryagin's Minimum Principle by increasing the collocation points and the mesh tolerance used in the simulation.

The fifth solution method utilizes the Tripath toolkit developed by Kallmann. The algorithm is built in C++ and accessed through a Python script run on a Linux operating system.

The data required to initialize the Tripath algorithm consists of the points of each closed polygonal constraint, the initial and final starting points of the path, and the offset distance required from each constrained edge. In order to achieve a feasible path for a SUAS, the offset distance was set to the minimum turning radius of the aircraft. The Tripath algorithm returns a Dubin's path solution with computation time on the order of milliseconds. The discretized solution that is returned is shown in Figure 7.

Here the Dubins path solution can be seen as a series of straight lines and constant radius turns. Given the position coordinates, and a solution satisfying Pontryagins Minimum Principle, the heading and heading rate can be determined for the two dimensional case with simple SUAS dynamics.

The sixth simulation uses the Tripath solution as the initial guess to the NLP solver. The Tripath toolkit returns the discretized path solution, the constrained Delaunay triangulation, and the search corridor. Implementing this data into the triangulation method with GPOPS-II, an optimal solution is obtained

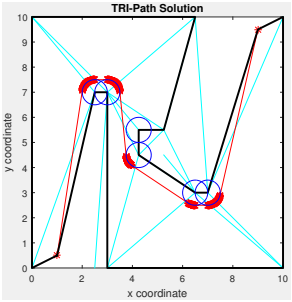

FIGURE 7. Tripath Toolkit Solution

and is presented through the given search corridor shown in Figure 8.

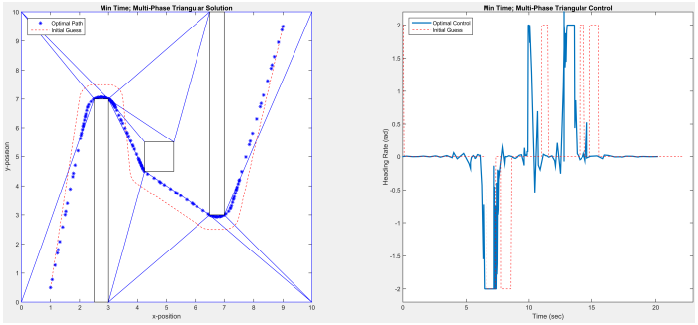

FIGURE 8. Hybrid Solution

The solution is optimal given the requirement that the path must traverse through each triangle of the search corridor in the same order as the initial guess. Again there can be seen some variance in the control due to the mesh tolerance setting of 10^{-2}. This allowed for a comparable number of collocation points to the other simulations.

RESULTS

For each simulation model, the mesh tolerance, computation time, objective time, collocation points, and the quality of the initial guess were recorded. The NLP solver in each of these simulation was "SNOPT". The results for each method are displayed in Table 1.

The Tripath toolkit solution is solved in 2.7 milli-seconds but also has the highest cost, does not include vehicle dynamics, and does not directly return the states and control of the vehicle, preventing the inclusion of more complex models. The hybrid solution solved with the fastest convergence time in GPOPS-II at 2.93 seconds with a performance index equivalent to the other optimal control methods.

Although the objective cost is compared to the first three methods, the problems that are solved differ by the way the

TABLE 1. Table of simulation results (using SNOPT)

Constraint	Mesh	Comp Time (s)	Obj Time (s)	Col Pts	Guess
Simple Shapes	10^{-4}	4.40	19.39	104	Good
Superellipse	10^{-5}	21.34	20.12	211	Perfect
Polygons	10^{-5}	9.96	20.68	159	Perfect
Triangulation	10^{-2}	17.37	20.14	221	Poor
Tripath	N/A	$2.7x10^{-3}$	22.6	377	N/A
Hybrid	10^{-2}	2.91	20.12	221	Tripath

constraints are formulated, resulting in a different search space. Therefore, the objective functions should not be compared directly, but rather used to assure each method is in close proximity to each other. The method solved with simple shape constraints returned the smallest value for the objective function, but did not adequately model a polygon constraint. The remaining methods, modeling constraints as polygons, each returned consistent values for the objective function.

In the case where constraints are composed of rounded edges, the triangulation method will require a polygon fit either to the interior or exterior of the rounded edge, dependent on the type of constraint being modeled. Soft constraints can be modeled to the interior of the rounded edge, allowing for the path to cut through small portions of the constraint. Hard constraints should be modeled on the exterior of the rounded edge where the model error will result in a longer path solution while eliminating the constraint from the solution space entirely. In either case, modeling errors can be minimized by increasing the vertices used to model the constraint, however each additional vertex requires additional phases in the optimal control problem. Further investigation is required to understand the impacts of large-phased problems.

CONCLUSION

The hybrid simplex method for optimal control was shown to be an efficient method for solving the minimum time SUAS optimal control problem through environments with arbitrary polygonal obstacles posed herein. By discretizing the space based on the constraints, a minimal number of triangles (phases) are required in the optimal control solution. Additionally, the path constraints are removed from the problem formulation in GPOPS-II and the transformation to the barycentric coordinate system standardizes the problem setup. The Tripath algorithm only requires the vertex points of each constraint, allowing the problem to be formulated and solved efficiently and with minimal information. From the analysis of the two-dimensional problem, computation times can reach levels feasible for on-board operations and therefore further evaluation of this methodology should be investigated to explore 6-DOF vehicle models and three-dimensional space.

ACKNOWLEDGMENT

The authors would like to thank David J. Grymin and the Air Force Research Laboratory, Aerospace Systems Directorate, Power and Controls Division, who sponsored the work and provided the challenging problem concept, background information, and continual support.

REFERENCES

[1] *Joint Concept of Operations for Unmanned Aircraft Systems*. US Department of Defense, Washington, D.C., 2011.

[2] Heidlauf, P. T., 2017. "Optimal UAV Path Planning with Dynamic No-Fly-Zones for Target Geolocation using Line-of-Bearing Measurements and Kalman Filtering". Thesis, Air Force Institute of Technology.

[3] Smith, N. E., 2014. "Optimal Collision Avoidance Trajectories for Unmanned/Remotely Piloted Aircraft". Dissertation, Air Force Institute of Technology.

[4] Humphreys, C. J., Cobb, R., Jacques, D. R., and Reeger, J. A., 2017. "Dynamic Re-plan of the Loyal Wingman Optimal Control Problem in a Changing Mission Environment". *AIAA Infotech @ Aerospace*(January), pp. 1–13.

[5] Masternak, T. J., 2014. "Multi-Objective Trajectory Optimization of a Hypersonic Reconnaissance Vehicle with Temperature Constraints". Dissertation, Air Force Institute of Technology.

[6] Carr, R. W., and Cobb, R., 2017. "An Energy Based Objective for Solving an Optimal Missile Evasion Problem". *AIAA Guidance, Navigation, and Control Conference*(88), pp. 1–18.

[7] Carr, R. W., and Lagimoniere, E., 2013. "A Range Safety Footprint Analysis for the Dream Chaser Engineering Test Article Using Trajectory Optimization". *AIAA Guidance, Navigation, and Control (GNC) Conference*.

[8] Benson, D., Huntington, G., Thorvaldsen, T., and Rao, A., 2006. "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method". *Journal of Guidance, Control, and Dynamics,* ***29***(6), pp. 1435–1440.

[9] Huntington, G., 2007. "Advancement and analysis of a Gauss pseudospectral transcription for optimal control problems". Dissertation, Massachusetts Institute of Technology.

[10] Miller, J. K., Llanos, P. J., and Hintz, G. R., 2014. "Optimal Control Framework for Impulsive Missile Interception Guidance". In Guidance, Navigation, and Control Conference, no. 8, pp. 1–14.

[11] Betts, J. T., 2008. "Practical methods for optimal control and estimation using nonlinear programming". *Advances in design and control*, p. 434.

[12] Suplisson, A. W., 2015. "Optimal Recovery Trajectories for Automatic Ground Collision Avoidance Systems (Auto GCAS)". Dissertation, Air Force Institute of Technology.

[13] Garg, D., 2011. "Advances In Global Psuedospectral Methods For Optimal Control". Dissertation, Florida.

[14] Burden, R. L., and Faires, J. D., 2010. *Numerical Analysis*.

[15] Trefethen, L. N., 2000. "Spectral Methods in Matlab". *Lloydia Cincinnati,* ***10***, p. 184.

[16] Jodeh, N. M., 2015. "Optimal UAS Assignments and Trajectories for Persistent Surveillance and Data Collection from a Wireless Sensor Network". Dissertation, Air Force Institute of Technology.

[17] Jaklic, Ales; Leonardo, Ales; Solina, F., 2013. *Segmentation and Recovery of Superquadrics*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

[18] Lewis, L. R., 2006. "Rapid motion planning and autonomous obstacle avoidance for unmanned vehicles".

[19] Hurni, M., 2009. "An information-centric approach to autonomous trajectory planning utilizing optimal control techniques". p. 296.

[20] Mohan, K., 2011. "Optimal Path Planning and Trajectory Optimization for". *University of Florida*, pp. 1–85.

[21] Shimrat, M., 1962. "Position of Point Relative to Polygon". *Communications of the ACM,* ***5***(8), aug, p. 434.

[22] Hacker, R., 1962. "Certification of Algorithm 112". *Communications of the ACM,* ***5***(12), dec, p. 606.

[23] Kapadia, M., and Badler, N. I., 2013. "Navigation and steering for autonomous virtual humans". *Wiley Interdisciplinary Reviews: Cognitive Science,* ***4***(3), pp. 263–272.

[24] Kallmann, Marcelo; Kapadia, M., 2016. *Geometric and Discrete Path Planning for Interactive Virtual Worlds*. Morgan & Claypool Publishers series.

[25] Juarez-Perez, A., and Kallmann, M., 2016. "Full-body behavioral path planning in cluttered environments". *Proceedings of the 9th International Conference on Motion in Games - MIG '16*, pp. 107–112.

[26] Chew, P. L., 1987. "Constrained Delaunay Triangulations". In Proceedings of the third annual symposium on Computational Geometry, pp. 215–222.

[27] Demyen, D., and Buro, M., 2006. "Efficient Triangulation-Based Pathfinding". In Proceedings of the 21st National Conference on Artificial Intelligence, AAAI, AAAI Press, pp. 942–947.

[28] Dechter, Rina; Judea, P., 1985. "Generalized best-first search strategies and the optimality of A*". *Journal of the ACM,* ***32***(3), pp. 505–536.

[29] Hetland, M. L., 2014. *Python Algorithms: Mastering Basic Algorithms in the Python Language*. Apress.

[30] Kallmann, M., 2010. "Shortest Paths with Arbitrary Clearance from Navigation Meshes". *Proceedings of the Eurographics SIGGRAPH Symposium on Computer Animation SCA*, pp. 159—-168.

[31] Kallmann, M., 2014. "Dynamic and Robust Local Clearance Triangulations". *Acm Transactions on Graphics,* ***33***(5), p. 17.

[32] Kallmann, M., and Kapadia, M., 2014. "Navigation meshes and real-time dynamic planning for virtual worlds". *ACM SIGGRAPH 2014 Courses*, pp. 1–81.

[33] Lee, D., and Preparata, F., 1984. "Euclidean shortest paths in the presence of rectilinear barriers". In Networks 14, pp. 393–410.

[34] Chazelle, B., 1982. "A theorem on polygon cutting with applications". In 23rd IEEE Symposium on Foundations of Computer Science, pp. 339–349.

[35] Hershberger, J., and Snoeyink, J., 1993. "Computing minimum length paths of a given homotopy class". *Computational Geometry: Theory and Applications,* ***4***(2), pp. 63–97.

[36] Kallmann, M., 2016. "Flexible and Efficient Navigation Meshes for Virtual Worlds Flexible and Efficient Navigation Meshes". In *Simulating Heterogenous Crowds iwth Interactive Behaviors*.

[37] Xu, J., and Güting, R. H., 2015. "Querying visible points in large obstructed space". *GeoInformatica,* ***19***(3), pp. 435–461.

[38] Brass, Peter; Vigan, Ivo; Xu, N., 2015. "Shortest path planning or a tethered robot". *Computational Geometry,* ***48***(9), pp. 732–742.

[39] Tanner, M. M., Burdick, J. W., and Nesnas, I. A. D., 2013. "Online motion planning for tethered robots in extreme terrain". *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5557–5564.

[40] Kallmann, M., 2010. "Navigation Queries from Triangular Meshes". *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* ***6459 LNCS***, pp. 230–241.

[41] Warren, J., Schaefer, S., Hirani, A. N., and Desbrun, M., 2007. "Barycentric coordinates for convex sets". *Advances in Computational Mathematics,* ***27***(3), pp. 319–338.

[42] Schindler, M., and Chen, E., 2012. "Barycentric Coordinates in Olympiad Geometry". pp. 1–40.

[43] Visser, T., De Visser, C. C., and Van Kampen, E.-J., 2015. "Quadrotor System Identification Using the Multivariate Multiplex B-Spline". *AIAA Atmospheric Flight Mechanics Conference*(January), pp. 1–13.

[44] Meyer, M., Barr, A., Lee, H., and Desbrun, M., 2002. "Generalized Barycentric Coordinates on Irregular Polygons". *Journal of Graphics Tools,* ***7***(1), pp. 13–22.

[45] Floater, M. S., 2003. "Mean value coordinates". *Computer Aided Geometric Design,* ***20***(1), pp. 19–27.

[46] Patterson, M. a., and Rao, A. V., 2010. "GPOPS II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hpAdaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming". *ACM Transactions on Mathematical Software,* ***41***(1), pp. 1–39.

[47] Betts, J. T., 1990. "Sparse Jacobian updates in the collocation method for optimal control problems". *Journal of Guidance, Control, and Dynamics,* ***13***(3), pp. 409–415.

Appendix C. SciTech Information Systems Conference 2018

Appendix C contains the second paper published in this research effort. This work contains the feasibility approach for the simplex optimal control problem in the constrained urban environment of downtown Chicago, USA. It was published and presented at the AIAA SciTech Information Systems Conference, Orlando, Florida, in January 2018.

AIAA SciTech Forum
8–12 January 2018, Kissimmee, Florida
2018 AIAA Information Systems-AIAA Infotech @ Aerospace

10.2514/6.2018-2259

Simplex Optimal Control Methods for Urban Environment Path Planning

Michael D. Zollars* and Richard G. Cobb†
Air Force Institute of Technology

David J. Grymin‡
Control Science Center of Excellence, Air Vehicles Directorate, Air Force Research Laboratories
Wright-Patterson Air Force Base, OH 45433

The work herein develops a feasible path solution through an urban environment for a small unmanned aerial vehicle with direct orthogonal collocation methods while leveraging navigation mesh techniques used in fast geometric path planners. Constrained optimal control problems for Small Unmanned Aircraft Systems (SUAS) have long suffered from excessive computation times caused by a combination of constraint modeling techniques and the quality of the initial path solution provided to the optimal control solver, ultimately preventing implementation into real-time, onboard systems. These issues are addressed herein by triangulating the search space to define a polygonal search corridor free of constraints while alleviating the dependency of defining problem specific parameters in the optimal control software. Utilizing algorithms developed for path planning of autonomous agents for computer animation, a search corridor is defined and an initial path solution is determined to initiate the optimal control problem. Results are applied to illustrate two-dimensional flight trajectories through downtown Chicago at an altitude of 550 feet Above Ground Level (AGL). Objective and computation times are reported to illustrate the feasibility of constructing and implementing the optimal control problem for onboard, real-time operations.

Nomenclature

α	Barycentric weights
Q	Matrix of polygon points
A	Matrix of barycentric weights
q	Simplex vertices in barycentric coordinates
δ	Building offset distance
r	Simplex vertices in Cartesian coordinates
R	Simplex interior point Cartesian coordinates
x	Position along the x-axis (ft)
y	Position along the y-axis (ft)
$\dot{x}$	Velocity in the x-axis (ft/s)
$\dot{y}$	Velocity in the y-axis (ft/s)
v	UAS airspeed (ft/s)
θ	UAS heading angle (rad)
$\dot{\theta}$	UAS heading angle rate (rad/s)
$\tilde{r}$	Minimum turn radius (ft)
X	State vector
J	Cost functional
P	Total number of phases
$\mathbf{u}$	Control vector
Superscript	
p	Phase number
Subscript	
0	Initial
f	Final

*PhD Candidate, Department of Aeronautics and Astronautics, 2950 Hobson Way, Maj, USAF, Member AIAA
†Professor, Department of Aeronautics and Astronautics, 2950 Hobson Way, USAF, Associate Fellow AIAA
‡Controls Research Engineer, Air Vehicles Directorate, AFRL, WPAFB, OH 45433

I. Introduction

In the discipline of optimization and control, path planning techniques have been used over a wide range of applications to find optimal flight trajectories through constrained environments. These path constraints may consist of terrain, infrastructure, or no-fly zones that must be avoided for the safety of the aircraft or the concealment of the air mission. Often, Small Unmanned Aerial Systems (SUAS) are teamed with manned aircraft to minimize the dangers the aircrew may encounter due to the constraints.[1] In the case of target localization, a host aircraft can remain at a safe altitude while a SUAS is deployed into a threat region to acquire and verify target coordinates. Once the initial target coordinates have been verified, the SUAS is required to determine a feasible path to a second target region. The autonomous localization of the target has been studied previously[2] while this paper focuses on the autonomous path development from the initial target region to the second target region through a constrained field.

In previous work studying constrained optimal flight trajectories,[3–7] significant issues have been illustrated that prevent the algorithm from reaching computation speeds required for real-time, onboard operations. For numerical solutions, an initial guess must be provided to the Non-Linear Program (NLP) solver and problem specific parameters must be estimated before a solution can be determined. The accuracy of these inputs will affect both the computational speed and the convergence to an optimal solution. This paper focuses on increasing computational speed of the optimal control solver and demonstrating convergence by eliminating constraints from the search space through triangulation and performing a coordinate transformation to standardize the input parameters to the NLP solver.

II. Background

II.A. Optimal Control

Due to the complexity of most optimal control problems, analytical solutions cannot be obtained and therefore numerical methods are employed to reach feasible solutions. Two common numerical approaches are described as the indirect method and the direct method. Indirect methods apply the calculus of variation to transform the optimal control problem into a Hamiltonian Boundary-Value Problem (HBVP). This method produces a highly accurate solution and assures the first-order optimality conditions are solved. However some disadvantages include a small radii of convergence and the requirement to analytically derive the HBVP. A good guess for the states, control and co-states is required for convergence, which often becomes time consuming and problematic, specifically for the co-states which often have no obvious physical meaning to the problem.[8]

To avoid these issues, direct methods transcribe the infinite-dimensional optimal control problem into a finite optimal control problem with algebraic constraints, otherwise known as an NLP.[9] Methods developed to transcribe the optimal control problem include direct shooting methods,[10] state and control parameterization methods,[11] and direct orthogonal collocation methods.[3,12] The focus of this research is on direct orthogonal collocation methods, also referred to as pseudospectral methods in the Aerospace field of study.[9]

When solving an optimal control problem with direct orthogonal collocation, the continuous time optimal control problem is transcribed to a discretized nonlinear programming problem. This is accomplished with three main concepts; orthogonal collocation, polynomial approximation, and Guassian quadrature.[13] For this research, the continuous functions of the optimal control problem are approximated with a finite dimensional Lagrange polynomial basis.[14] The state x is approximated at a set of collocation points described as

$$\tilde{x}(\tau) \approx \tilde{x}_N(\tau) = \sum_{i=1}^{n+1} x_i L_i(\tau) \tag{1}$$

where x_i represents the weight function, $L_i(\tau)$ is the Lagrange polynomial basis and τ represents an affine transformation of the time t on the interval from $(-1, 1)$.

With the problem discretized, Gaussian quadrature is used for differentiation or integration of the state and control. This method is termed a global method as each collocation point is solved simultaneously rather than other fixed interval methods such as a 3 or 5 point formula method.[15] The error produced by this method can be greatly reduced by choosing the collocation points appropriately with Legendre or Chebyshev point placement to minimize the affects of Runge phenomenon.

II.B. Constraint Models

When defining the optimal control problem, many techniques have been employed to appropriately model path constraints in two-dimensional space. One of the most commonly employed techniques is to simulate a vehicle no-fly zone with circular or spherical constraint function[5, 12, 16] of the form

$$\|x(t) - \tilde{x}\|_2 > r_n. \tag{2}$$

Here, the difference between the position states of the vehicle, $x(t)$ and the center of the no fly-zone, $\tilde{x}$ must be greater than a defined radius, r_n. These functions are differentiable and smooth and therefore can be handled with relative ease with an NLP solver such as *IPOPT* or *SNOPT*. The drawback is that the constraint is restricted to circular or spherical shapes which may not adequately represent the true constrained environment.

Superquadrics have been used extensively in computer vision, computer graphics, and robotics[17] to extend circular constraints to shapes resembling squares and rectangles. The superellipse is a type of Leme curve which is defined as

$$\left(\frac{x_c}{a}\right)^m + \left(\frac{y_c}{b}\right)^m = 1 \tag{3}$$

where the semi-major and semi-minor axes are defined by a and b, x_c and y_c represent the center point of the object, and m is a rational number defining the curvature of the object. When solving problems with direct orthogonal collocation, the superquadric has been used to include constraint regions other than simple geometric shapes. Hurni, Lewis and Mohan used the superquadric in a constraint rich environment and found success in convergence but at the cost of computation time.[18–20]

To further extend the constraint model, the ray-casting algorithm, used to determine if a point is inside an arbitrary polygon, returns a boolean variable, in this case with `True` corresponding to a point outside a polygon, and `False` for a point inside a polygon. To eliminate the discontinuity along the edges of the polygons, the result of the ray-casting algorithm is then multiplied by the distance to the nearest polygon.

Given a line, denoted as $\mathbf{v}$, passing through the points (x_1, y_1) and (x_2, y_2), and an arbitrary point denoted as $\mathbf{p} = (x_p, y_p)$, let $\mathbf{n} = (x_n, y_n)$ denote the point on line $\mathbf{v}$ nearest to the point $\mathbf{p}$. The point $\mathbf{n}$ is computed as below.

$$u = \frac{(x_p - x_1)(x_2 - x_1) + (y_p - y_1)(y_2 - y_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4}$$

$$x_n = x_1 + u(x_2 - x_1) \tag{5}$$

$$y_n = y_1 + u(y_2 - y_1) \tag{6}$$

If $\mathbf{v}$ is a line segment, $u \in [0, 1]$. Therefore if $u < 0$ according to Eq. (4), $u = 0$ and if $u > 1$, $u = 1$. The distance from $\mathbf{p}$ to $\mathbf{n}$ then gives the minimum distance from $\mathbf{p}$ to $\mathbf{v}$.

To compute the shortest distance to any edge in the environment, the above procedure for finding the minimum distance to a segment is computed for all segments of all polygons in the environment. The minimum distance for the environment, d_{env}, is set to an arbitrarily high value. Then, for each segment, the distance d_i is computed. If $d_i < d_{\text{env}}$, then $d_{\text{env}} = d_i$. Since d_{env} only computes the minimum distance to an edge of a polygon in the environment, it is then multiplied by the ray-casting result, computed in MATLAB using the `inpolygon` function. The result returns a value of $d_{\text{env}} = 0$ for all points that are inside a polygon, and $d_{\text{env}} > 0$ for all points outside a polygon.

Each of these methods are based on the premise that the constraint is contained inside the search space and the NLP solver must evaluate constraint equations to assure a feasible solution can be attained. Previous work has shown the comparison and limitations of each of these methods.[21] The following section leverages research from the field of computer animation to eliminate the constraints from the search space by performing a triangular discretization dependent on the constraint field rather than the size of the search space.

II.C. Fast Geometric Path Planning

Direct orthogonal collocation methods are capable of meeting the computational requirements for onboard SUAS operations if an adequate starting point of the states are given to the NLP solver. When approaching the constant altitude SUAS problem, comparisons can be made to path development of autonomous agents

in the interactive virtual world. Virtual worlds are populated with autonomous virtual humans, or agents, where computed paths must take into account path length, time, and energy expended traversing the path.[22]

The demand on the complexity of the virtual world combined with improved graphic capability has encouraged new techniques for achieving intelligent navigation for the next generation virtual agent simulation.[23] Given the speeds of which these video games are played, path planning algorithms must perform efficiently under limited time budgets to traverse the autonomous agent from one end of a building to the other. Often greedy algorithms are chosen in which the first feasible path that is found is continuously improved according to the characteristic dynamics and a pre-determined threshold on computational time. The path available at the time required is then implemented.[22,24] This can often result in a sub-optimal path, but in the virtual world of the gaming industry, the timeliness of the solution is more heavily favored over the most optimal route.

II.D. Constrained Deluanay Triangulation

Often, fast geometric path planning algorithms discretize the search space with a triangular mesh formed with a Constrained Delaunay Triangulation (CDT). A CDT is a refinement of the Delaunay triangulation that forces a required segment as an edge of the triangulation. The formal definition is described as follows:[25]

Definition. Let G be a straight-line constrained edge. A triangulation T is a CDT of G if each edge of G is also an edge of T and for each remaining edge e of T there exists a circle C with the following properties:

1. The endpoints of edge e are on the boundary of C.
2. If any vertex v of G is in the interior of C then it cannot be "seen" from at least one of the endpoints of e (i.e., if you draw the line segments from v to each endpoint of e then at least one of the line segments crosses an edge of G).

The CDT is made of constrained edges and unconstrained edges. When determining an optimal path, the solution can cross an unconstrained edge, but must avoid all constrained edges. Since the constrained edges are required, often formulation of the CDT contains a vertex or an edge that does not satisfy the Delaunay triangulation conditions, and therefore a CDT may not be equivalent to a Delaunay triangulation. An example of the differences can be seen below in Figure 1.

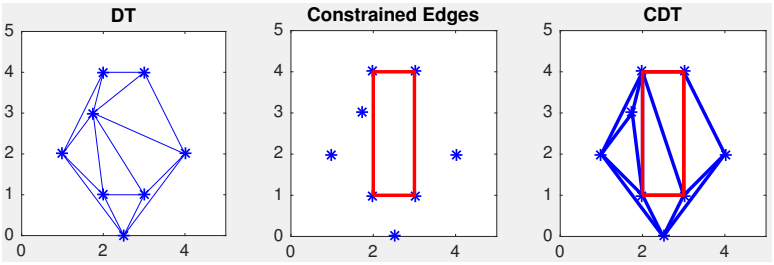

Figure 1. Left: Unconstrained Delaunay Triangulation; Middle: Constrained Edge Segments; Right: Constrained Delaunay Triangulation[25]

Here, the left image of Figure 1 shows a Delaunay triangulation given a set a points in the search field. The middle image displays a four sided polygon representing a path constraint such as a building or no fly zone. The right image illustrates the CDT with the constrained edges included in the triangulation.

Using this technique, constraints can be forced into the discretization of the space. In computer animation, these constraints represent walls in buildings, tables, chairs, and other common structures autonomous agents are required to navigate around. Applying these constraints to the SUAS path planning problem, these boundaries illustrate hard constraint no-fly zones that are represented with polygonal shapes such as buildings, terrain, and restricted airspace.

III. Triangulation-Based Path Development

Path planning algorithms have been designed to determine the best route from an initial starting point to a goal location while satisfying conditions on constraints and dynamics. In the field of optimal control, the dynamics of the agent must be satisfied while obeying all constraints placed on the states, control, and

path. In computer animation, path planning focuses on determining the best route for the autonomous agent while avoiding walls, furniture and other obstacles. Many algorithms have been developed to accomplish autonomous agent path planning while placing an emphasis on computational speed as solutions must be found quickly to give the appearance the agent is in fact making real-time intelligent decisions as the video game progresses.[26]

III.A. A* Search Algorithm

The A* search algorithm gets its roots from Dynamic Programming and is an extension to Dijkstra's algorithm. The entire domain is searched with Dijkstra's algorithm, whereas in an A* search is guided by a heuristic. It is considered an "informed search" or "best-first search" as the algorithm searches the space until a solution is found. This solution may be sub-optimal and therefore the algorithm will continue to search the space, updating the solution if a lower performance index is found. Often a time limit is imposed on the algorithm and sub-optimal paths are accepted in order to find a solution in the quickest possible manner. In computer animation, this is a trade-off in order to achieve the computational times required for simulation.

The initial node for the A* algorithm is given a cost value of $f(1) = 0$ and additional nodes, (n), are searched in a tree like fashion. A heuristic is introduced and added to the cost function to give a priority in the direction of the search. Often the Euclidean distance is used as the heuristic and the overall cost function is

$$f(n) = g(n) + h(n) \tag{7}$$

where $g(n)$ is the cost evaluated at the current node and $h(n)$ is a heuristic that estimates the cheapest path from the current node to the endpoint.[27] At each iteration, the A* algorithm must determine which nodes to expand by selecting the path that minimizes the current cost, $g(n)$ and the cost to go, $h(n)$. In the case where $h(n)$ is set equal to 0, the result will be consistent with Dijkstra's algorithm.[28]

Disadvantages of the A* algorithm are related to the choice of heuristic which must be chosen to underestimate the cost, thus giving a lower bound to the solution. This may cause the algorithm to spend computation time discriminating between two paths of equal remaining distances. Additionally, the result produces an optimistic *estimate* of all possible solutions, resulting in a path that may be sub-optimal if the algorithm is terminated after the goal node is first reached.[27]

III.B. Local Clearance Triangulation

Marcello Kallmann has provided an extensive review of path development with clearances[29] and developed the Triplanner toolkit algorithms designed to determine the shortest path while accounting for minimum clearance distances to all constraints.[30] Specifically, he focused on determining paths of shortest distances, efficiency of computation time, and maintaining proper clearance from obstacles along the calculated path. An overview of the development of the 2010 Triplanner algorithm used herein is discussed below while a more extensive review of the algorithm can be found in Kallmann's work.[29,31]

First, let S define a set of n segments that form all the constrained edges in the domain. The set of all endpoints of each segment forms an $n \times 2$ matrix P. A CDT, T, is then formed such that all segments of S are also segments of T and the Delaunay criterion are upheld. Next, to account for the width of the autonomous agent, Kallmann performs a test to assure that a disk of radius r can traverse through any given region. This enables an efficient computation of paths with arbitrary clearances. To assure the accuracy of the results, a local clearance test is conducted to guarantee a solution will be contained in a path with a minimum radius of $2r$. If the test fails, a refinement of the mesh is performed by redistributing the triangulation or adding a vertex point to a straight line segment of the set S. This may result in a new set of constrained edges, as a straight line segment could be subdivided into multiple sections. Once the triangulation has passed the local clearance test, the final mesh is termed a "Local Clearance Triangulation (LCT)".

A path through the LCT from a starting point p to a finish point q is defined as "free" if it does not cross a constrained edge. A free path will cross several unconstrained edges resulting in a "channel" formed of all traversed triangles. Using an A* search algorithm in a discretized space formed by the triangulation, a geometric solution for the channel is determined. The channel formed by the A* search is dependent on the chosen cost function and will result in a shortest path search defined by the cost metric. A simplistic cost metric may consist of using the centroid of each triangle or instead using the midpoint of each unconstrained

segment to determine the best channel. Kallmann settled on a cost function which begins the search from the midpoint of each triangle, but enhanced the search through a reference point connecting the point in the previously traversed edge to the final point q. If q is not visible on the straight line path, the nearest vertex is used for the traversed point while accounting for the radial offset distance form each vertex point.

After the channel is determined, a funnel algorithm is used to find the shortest path within the channel. The funnel algorithm was developed by Lee and Preparata, and Chazelle[32,33] as cited by Hershberger.[34] The algorithm has been used to calculate the shortest path under multiple applications; including path finding for autonomous agents,[35] querying visible points in large data sets to define shortest paths,[36] shortest paths for tethered robots[37] and robots in extreme terrain.[38]

Given a corridor with a starting position p and final point q, the funnel algorithm defines the entrance point of the simplex as the apex, a, and the starting point in the funnel. Since the corridor has already been defined using the A* algorithm, the two vertices connecting the next simplex can be defined as u and v, with the third vertex w. If the straight line solution from a to w is feasible, a straight line path is chosen from a to w as shown in Figure 2A. Maintaining a as the apex, the third vertex of the next simplex is evaluated for the straight line path solution from a to w'as shown in Figure 2B. In the case where the straight line solution is not feasible, the vertex closest to the next point in the path is chosen as the new apex, a' and the algorithm continues as shown in Figure 2C. Further detailed explanation of the funnel algorithm can be found in Hershberger's work.[34] To deal with the local clearance around obstacles, Kallmann implemented a required circular constraint of radius r on each included vertex as illustrated in Figure 2D.[39]

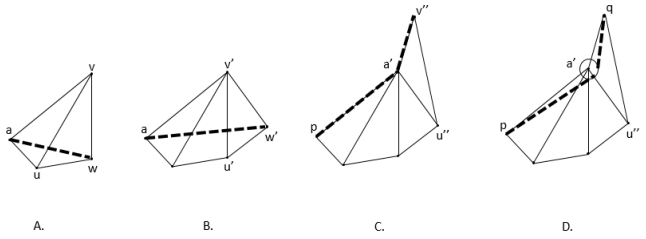

Figure 2. Implementation of the Funnel Algorithm on a Simplex Corridor[40]

The Triplanner toolkit performs a locally optimal search and is capable of achieving path solutions on the order of milliseconds for environments with 60K+ segments.[29] Although there is no guarantee that the solution is the global optimal path, the simplex corridor found and searched is free of constraints allowing for solutions consisting of straight line segments and max turn radius arcs.

III.C. Barycentric Coordinate Frame

Often when dealing with simplex shapes, the barycentric coordinate frame is preferred over traditional Cartesian coordinates. A barycentric coordinate system defines the location of a point within a simplex as a weighted measure to each of the vertex points, also referred to as areal coordinates in the context of triangles.[41] The barycentric coordinate system defines the side of the simplex as the axes, allowing for simple representations of lines, points, and perpendicular relationships.[42]

To define the coordinate system, let r_1, r_2, ..., r_n be n vertices of a complex planar polygon Q in $\mathbb{R}^2$ for $n \geq 3$. For the purposes of this research, when investigating problems in $\mathbb{R}^2$, triangulation techniques will be utilized and therefore $n = 3$. Any point, R, inside polygon Q can be represented in barycentric coordinates defined with the vertices of Q used as a basis as follows:[43–45]

$$\mathbf{R} = \sum_{j=1}^{n} \alpha_j \mathbf{r_j} \tag{8}$$

where α represents the barycentric weights, defined as a set of real coefficients whose added sum equals unity,

$$\sum_{j=1}^{n} \alpha_j = 1. \tag{9}$$

To ensure that each point remains inside the polygon, Q,

$$\alpha_j \geq 0 \ \ \forall j \in [1...n]. \tag{10}$$

For the triangular relationship, $n = 3$, transformation from a barycentric coordinate frame to a Cartesian coordinate frame can be easily performed through a linear transformation of the coordinates represented as:

$$\mathbf{R} = \mathbf{QA} \tag{11}$$

where $\mathbf{R} \in \mathbb{R}^2$ defines the point location in the Cartesian coordinate frame, $\mathbf{Q} \in \mathbb{R}^{2\times n}$ represents the matrix of vertices in the polygon, and $\mathbf{A} \in \mathbb{R}^{\mathbf{n}}$ describes the weight matrix and the set of barycentric coordinates.

III.D. Coordinate Transformation

In order to simplify the optimal control solver input parameters, the problem is expressed in the barycentric coordinate system. Given a simplex shown in Figure 3, each vertex is defined in Cartesian coordinates as

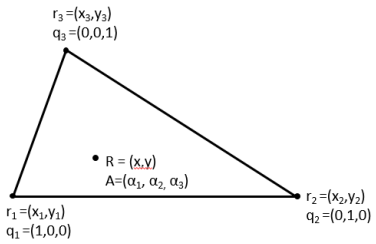

Figure 3. Barycentric Coordinate Frame

$$\mathbf{r_i} = (x_i, y_i) \ \ \forall i \in [1...n] \tag{12}$$

where n is equal to the number of sides of the simplex. The solution is limited to the constant altitude, two-dimensional plane and therefore $n = 3$. Each point within the simplex can be represented as an ordered triple of real numbers, representing the weighted distribution to each vertex. Each vertex is defined in barycentric coordinates as:

$$\mathbf{q_1} = (\mathbf{1}, \mathbf{0}, \mathbf{0}) \tag{13}$$

$$\mathbf{q_2} = (\mathbf{0}, \mathbf{1}, \mathbf{0}) \tag{14}$$

$$\mathbf{q_3} = (\mathbf{0}, \mathbf{0}, \mathbf{1}) \tag{15}$$

while any point within the simplex is represented with the corresponding weights to each vertex

$$\mathbf{A} = (\alpha_\mathbf{1}, \alpha_\mathbf{2}, \alpha_\mathbf{3}). \tag{16}$$

Expanding Equation 8 and representing the final weight α_3 in terms of the first two weights, the weights of the barycentric coordinate frame are expressed in terms of the Cartesian coordinates,

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = T^{-1}(\mathbf{R} - \mathbf{r_3}) \tag{17}$$

$$\alpha_3 = 1 - \alpha_1 - \alpha_2 \tag{18}$$

where T is a $2x2$ matrix representing the vertices of the triangle as

$$T = \begin{pmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{pmatrix}, \tag{19}$$

and $\mathbf{R} - \mathbf{r_3}$ is a $2x1$ vector summation of the Cartesian coordinates. Expanding Equation 17 gives the barycentric weights in terms of the vertex coordinates and the point location within the triangle.

$$\alpha_1 = \frac{(y_2-y_3)(x-x_3)+(x_3-x_2)(y-y_3)}{det(T)} \tag{20}$$

$$\alpha_2 = \frac{(y_3-y_1)(x-x_3)+(x_1-x_3)(y-y_3)}{det(T)} \tag{21}$$

$$\alpha_3 = 1 - \alpha_1 - \alpha_2 \tag{22}$$

Differentiating the weights with respect to the x and y position gives the following set of dynamic equations

$$\dot{\alpha}_1 = \frac{(y_2-y_3)\dot{x}+(x_3-x_2)\dot{y}}{det(T)} \tag{23}$$

$$\dot{\alpha}_1 = \frac{(y_3-y_1)\dot{x}+(x_1-x_3)\dot{y}}{det(T)} \tag{24}$$

$$\dot{\alpha}_3 = -\dot{\alpha}_1 - \dot{\alpha}_2. \tag{25}$$

IV. Solution Methodology

Fast geometric path planning algorithms, such as the Triplanner toolkit, have been designed to formulate path solutions in constrained, two-dimensional environments on the order of milliseconds. The resulting Dubins path solution provides a feasible flight path through a chosen search corridor. The aircraft dynamics are accounted for by a radial offset distance from each constrained edge equal to the SUAS turn radius. The heading angle is determined by the angle difference of consecutive points while the heading rate and control are each derived with a two-point finite differencing method of the heading angle vector. To evaluate the optimality of the Triplanner path solution and its viability for use in SUAS control software, the problem is translated to the optimal control software, GPOPS-II where a comparison can be made to the optimal flight path, heading angle, heading rate, and control.

GPOPS-II is described as a computational tool for solving multiple-phase optimal control problems using variable-order Gaussian quadrature collocation methods.[46] Each phase is defined with a set of dynamic, path, integral, and parameter constraints. Phases are linked through event constraints that relate information at the start and terminal point of each phase and allow for independent variables to be continuously transitioned through each phase.[11,47]

With this program, a solution through one simplex can be represented as one phase in GPOPS-II. Formulating the optimal control problem through a CDT provides the basis for a trajectory solution that traverses through a corridor of simplexes (as described in the previous section) each represented as a single phase, linked together through event constraints to determine the optimal solution. It is recognized here that the solution found by this method is dependent on the simplex corridor that is presented to the optimal solver from the Triplanner algorithm. This corridor may not provide the global optimal solution, rather a locally optimal solution which provides an admissible set satisfying the dynamic constraints. Supplying this corridor to the NLP solver, the simplex vertices are ordered in such a manner that the shared vertices of consecutive triangles are properly indexed to assure the state vector is properly read into the optimal control software, GPOPS-II. As the path trajectory traverses across a simplex edge, one of the weights (states $1-3$) will be zero as the weight associated with the opposing vertex has no contribution to the location of the point. As the new phase begins, it is imperative that the states of the next simplex match the states of the previous. In other words, the opposite vertex of the new simplex must accept the zero value and the associated weights for the other two vertices must match appropriately in the state vector. This is illustrated in Figure 4.

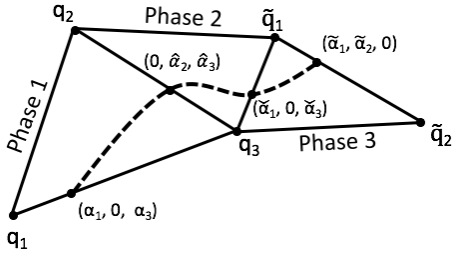

Figure 4. Simplex Phased Solution for Optimal Control

The following sections define the optimal control problem in terms of a phased approach. Each simplex will be represented in barycentric coordinates with appropriate dynamic constraints, path constraints, and parameter constraints. The number of phases required to be solved is problem specific and defined after the space has been discretized into a CDT. The optimal control solver will search for a path through a simplex corridor that is defined prior to initiating the optimal control problem. The total number of solution phases is represented with the variable P.

IV.A. SUAS Dynamics

The dynamics of the aircraft are formulated with a four state model, representing the position of the SUAS in the $x(t)$, $y(t)$ directions, the heading angle, $\theta(t)$, and the heading rate, $\dot{\theta}(t)$. The control for the aircraft is the derivative of the heading angle rate, $\ddot{\theta}(t)$. The SUAS dynamics are described below as a relationship between the states and the controls

$$\dot{x}^{(p)}(t) = (v)cos(\theta^{(p)}(t)) \ \ \forall p \in [1...P] \tag{26}$$
$$\dot{y}^{(p)}(t) = (v)sin(\theta^{(p)}(t)) \ \ \forall p \in [1...P] \tag{27}$$
$$\dot{\theta}^{(p)}(t) = \gamma(t) \ \ \forall p \in [1...P] \tag{28}$$
$$\ddot{\theta}^{(p)}(t) = u(t) \ \ \forall p \in [1...P] \tag{29}$$

where v represents the aircraft velocity which is held constant in this work at $30 ft/s$.

IV.B. Optimal Control Problem

By discretizing the search space with a CDT, a phased approach is taken in the optimal control problem. Each triangle is solved as a single phase in GPOPS-II and each phase is connected through event constraints. The dynamics constrain the path through each triangle in barycentric coordinates. Combining the described steps, the optimal control problem formulation has been consolidated as follows.

Minimize the cost functional

$$J^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} dt \ \ \forall p \in [1...P] \tag{30}$$
$$J = \sum_{p=1}^{P} J^{(p)} \tag{31}$$

subject to the dynamic constraints

$$\dot{\alpha}_1^{(p)}(t) = \frac{(y_2-y_3)(v)cos(\theta^{(p)}(t))+(x_3-x_2)(v)sin(\theta^{(p)}(t))}{det(T)} \ \ \forall p \in [1...P] \tag{32}$$
$$\dot{\alpha}_1^{(p)}(t) = \frac{(y_3-y_1)(v)cos(\theta^{(p)}(t))+(x_1-x_3)(v)sin(\theta^{(p)}(t))}{det(T)} \ \ \forall p \in [1...P] \tag{33}$$
$$\dot{\alpha}_3^{(p)}(t) = -\dot{\alpha}_1^{(p)}(t) - \dot{\alpha}_2^{(p)}(t) \ \ \forall p \in [1...P] \tag{34}$$
$$\dot{\theta}^{(p)}(t) = \gamma(t) \ \ \forall p \in [1...P] \tag{35}$$
$$\ddot{\theta}^{(p)}(t) = u(t) \ \ \forall p \in [1...P] \tag{36}$$

with the control

$$u^{(p)}(t) = \ddot{\theta}(t) \tag{37}$$

and the state vector defined as

$$\mathbf{X} = (\alpha_1, \alpha_2, \alpha_3, \theta, \dot{\theta}) \tag{38}$$

with boundary conditions given as the initial and final constraints,

$$\mathbf{X}^{(1)}(t_0^{(1)}) = ((\alpha_1)_0, (\alpha_2)_0, (\alpha_3)_0, (\theta)_0, (\dot{\theta})_0) \tag{39}$$
$$\mathbf{X}^{(P)}(t_f^{(P)}) = ((\alpha_1)_f, (\alpha_2)_f, (\alpha_3)_f, (\theta)_f, (\dot{\theta})_f). \tag{40}$$

Inequality path constraints representing bounds on the state, control and time are defined as

$$0 \leq \alpha_1^{(p)} \leq 1 \tag{41}$$
$$0 \leq \alpha_2^{(p)} \leq 1 \tag{42}$$
$$0 \leq \alpha_3^{(p)} \leq 1 \tag{43}$$
$$|\theta^{(p)}| \leq 180 \ \ deg \tag{44}$$
$$|\dot{\theta}^{(p)}| \leq 25 \ \ deg/s \tag{45}$$
$$|u^{(p)}| \leq 1 \ \ deg/s^2 \tag{46}$$
$$0 \leq t^{(p)} \leq \frac{edge_{max}^{(p)}}{v}, \tag{47}$$

where $edge_{max}$ represents the longest edge in the defined simplex. Equality path constraints are implemented to allow the SUAS to maintain a safe distance from each constrained edge of the simplex corridor,

$$\left(r_{1x}^{(p)} - x^{(p)}(t)\right)^2 + \left(r_{1y}^{(p)} - y^{(p)}(t)\right)^2 - (\delta)^2 = 0 \tag{48}$$

$$\left(r_{2x}^{(p)} - x^{(p)}(t)\right)^2 + \left(r_{2y}^{(p)} - y^{(p)}(t)\right)^2 - (\delta)^2 = 0 \tag{49}$$

$$\left(r_{3x}^{(p)} - x^{(p)}(t)\right)^2 + \left(r_{3y}^{(p)} - y^{(p)}(t)\right)^2 - (\delta)^2 = 0 \tag{50}$$

where r_i represents the x and y coordinate of simplex vertices and δ defines the minimum safety buffer between the SUAS and the building constraints defined by the distance between each vertex point and the flight path. Finally, event constraints are included to maintain a continuous transition of the state variables between each phase,

$$X_o^{(p+1)} - X_f^{(p)} = 0 \ \ \forall p \in [1...P-1]. \tag{51}$$

IV.C. Initial Guess to the NLP

To achieve computation times within the limits for onboard SUAS operations, a good initial guess is required for the NLP solver. The methodology presented discretizes the search space into a triangular mesh and a search corridor of connected simplexes defines the search space, free of constraints. This allows for an estimate of the optimal path for the first simulation to be comprised of a path solution that connects the midpoint of each unconstrained edge of the search corridor. The heading is initiated with a constant vector made up of the angles between the initial and final locations while the heading rate and the control are initiated with the zero vector. Finally, the time vector is estimated through each phase as the length of the longest simplex edge divided by the vehicle airspeed.

The output of the Triplanner toolkit solution consists of a discretized path solution within the defined search corridor that provides an initial guess to the second simulation. This path consists of straight line segments made up of the two end points connected to sections of constant radius turns. These points are interpolated to assure data points are available in each defined simplex. The barycentric coordinates are calculated for each simplex from the interpolated discretized path. The heading angle, θ, is determined by the angle between consecutive Cartesian coordinates of the path. The heading rate and control are calculated with a right point finite differencing method initiated with the heading angle vector. Both the heading rate and control are rate limited to be consistent with those used in the optimal control problem found in Equations 45 and 46. Finally, the time vector is calculated as the running summation of the Euclidean distance between each consecutive points divided by the speed of the aircraft.

V. Results

This section illustrates a solution for a single SUAS flying through downtown Chicago with an altitude constraint of $600ft$ AGL. The scenario is chosen to represent a challenging urban environment to verify the functionality of the solution method. The search space is defined by a small region of downtown Chicago measuring $5600ft$ by $2800ft$. All structures within this region that exceed $550ft$ AGL are modeled with a simplistic four-sided polygon. For this scenario, the initial starting point for the SUAS is a parking garage located on the south-west side of the city, while the final location is a monument located on the north-east side of the city. The initial and final aircraft constraints are defined as

$$(x_0, y_0, \theta_0) = (200, 200, free) \tag{52}$$

$$(x_f, y_f, \theta_f) = (4700, 2650, free). \tag{53}$$

Within the defined search space, there are 37 buildings that exceed $550ft$. Figure 5A shows the constraint map with the initial and final flight coordinates illustrated with a green and red asterisk respectfully. Performing a CDT on the space, Figure 5B illustrates the discretized mesh defined by the constrained environment. Each building taller than $550ft$ is represented on the map as a red, four sided polygon.

Utilizing the 2010 version of the Triplanner toolkit developed by Kallmann and his team, the search corridor is defined and a feasible flight path solution is determined. The algorithm is built in C++ and

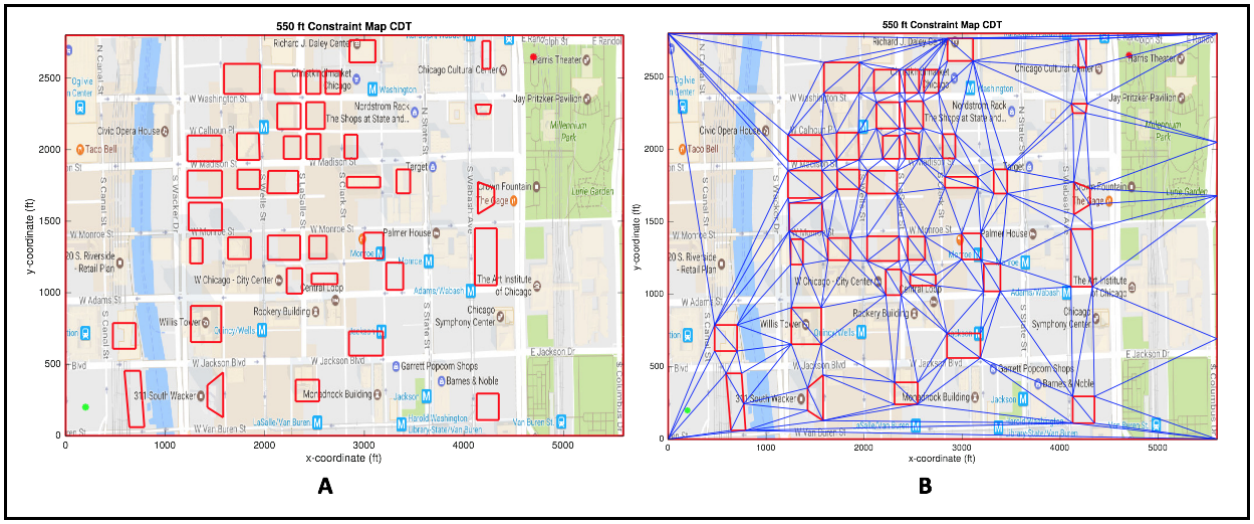

Figure 5. Downtown Chicago; Constraint Map (A), Discretized Mesh (B). Map data @2017 Google

accessed through a Python script run in a Linux operating environment. The data required to initialize the Triplanner algorithm consists of the points of each closed polygonal constraint, the initial and final starting points of the path, and the offset distance required from each constrained edge. In order to achieve a feasible path for a SUAS, the offset distance was set to the minimum turning radius of the aircraft determined through the relationship between the aircraft's velocity and turn rate as follows,

$$\tilde{r} = \frac{v}{\dot{\theta}_{min}} \tag{54}$$

for $\tilde{r}$ is the minimum turn radius, v is the velocity, and ω is the turn rate. The Triplanner algorithm returns a Dubins path solution with computation time on the order of milli-seconds. The search corridor and the path solution returned from the Triplanner toolkit is shown in Figure 6.

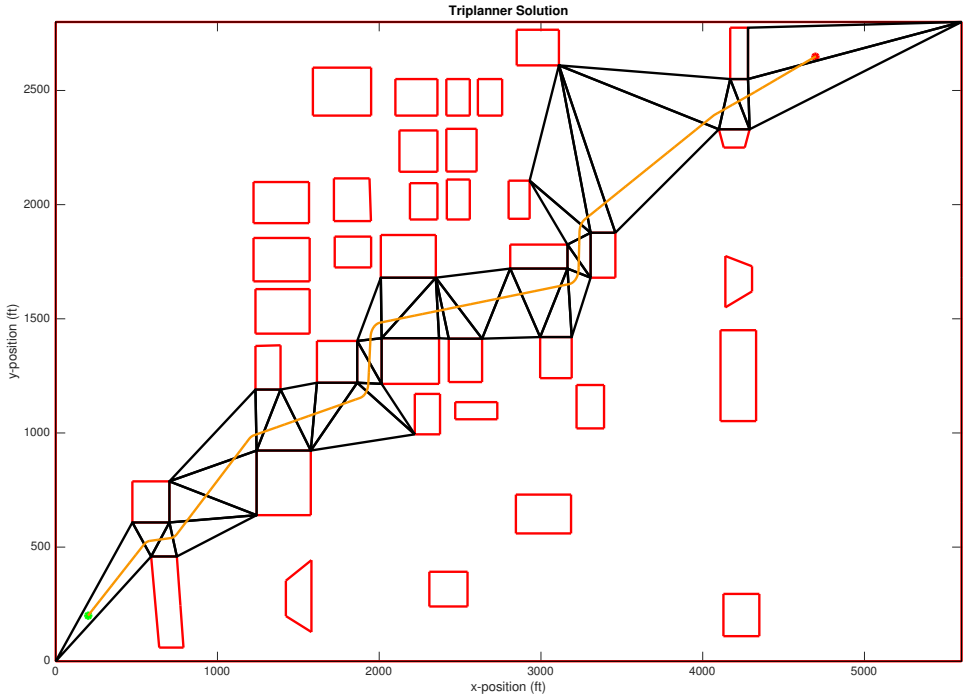

Figure 6. Triplanner Toolkit Solution. Map data @2017 Google

The Triplanner solution is solved in 5.84 milliseconds, with an objective cost of 177.8, however, the aircraft dynamics and control are not incorporated into the Triplanner toolkit as the Dubins path solution is found through a discretized A* search algorithm. The output of the Triplanner algorithm provides the triangulated mesh that includes all the constrained and unconstrained edges as well as the discretized path solution. This data is interpolated to provide a solution comprised of data points at $1ft$ spacing.

Two simulations were executed to illustrate the necessity of providing a quality guess to the optimal control software. For each simulation, the optimal control problem is solved in GPOPS-II under the same set up parameters with only the initial guess for the state vector, control, and time differing. The key GPOPS-II parameters are listed below in Table 1.

Table 1. GPOPS-II User Settings

GPOPS-II User Settings	
Mesh Method	hp-PattersonRao
Mesh Tolerance	10^{-2}
NLP Solver	SNOPT
Derivative Supplier	AdiGator
Method	RPM-differential
NLP Tolerance	10^{-3}
Min Collocation Points	4
Max Collocation Points	10
Mesh Fraction	$\frac{1}{2}$*ones(1,2)
Mesh Collocation Points	4*ones(1,4)

The minimum flight safety buffer, preventing an aircraft from flying to close to a building, δ, is set to $15ft$ for each simulation. The path results for the first simulation are shown in Figure 7. The initial guess

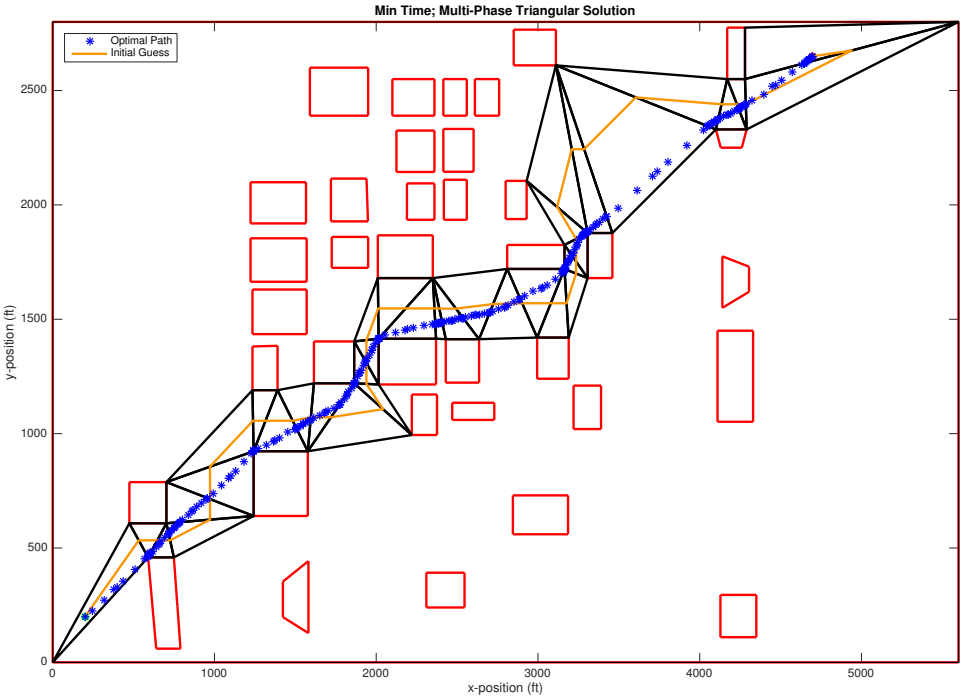

Figure 7. Optimal Solution with Simplex Mid-Point Guess. Map data @2017 Google

is illustrated with the dotted red line while the discretized path solution is shown with the blue asterisks. An optimal solution through the search corridor was found with a computation time of 4.57 seconds. The heading angle and the heading angle rate are the fourth and fifth states and are displayed in the top two plots of Figure 8. The control is shown in the lower plot of Figure 8 with an initial guess equal to the zero vector. Evaluating these plots, the mid-point solution requires a heading change at the start of each simplex. This requires excessive vehicle control inputs when compared to the the optimal solution. The second simulation

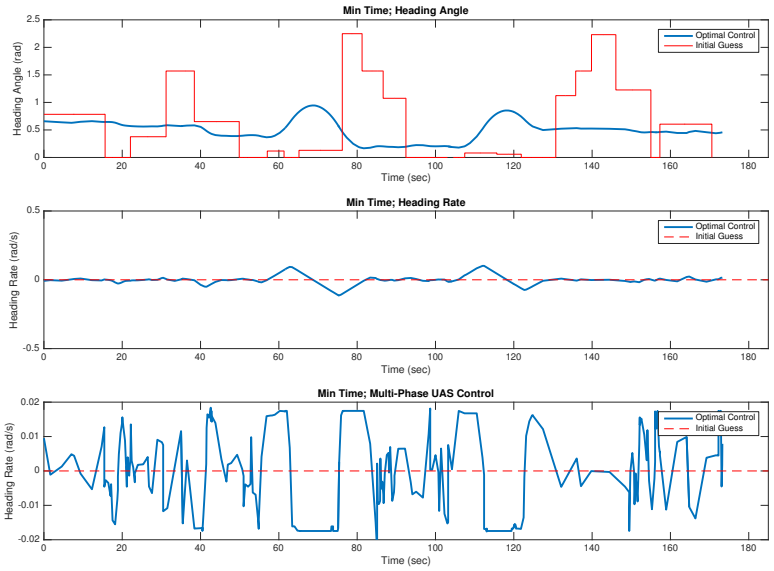

Figure 8. Mid-Point Guess Theta and Control Solution

provides an initial guess vector for the state, control, and time as determined for each triangle through the Triplanner toolkit solution and implemented in GPOPS-II as individual phases. The results are shown below in Figure 9. Again, the initial guess is illustrated with the dotted red line while the discretized path solution

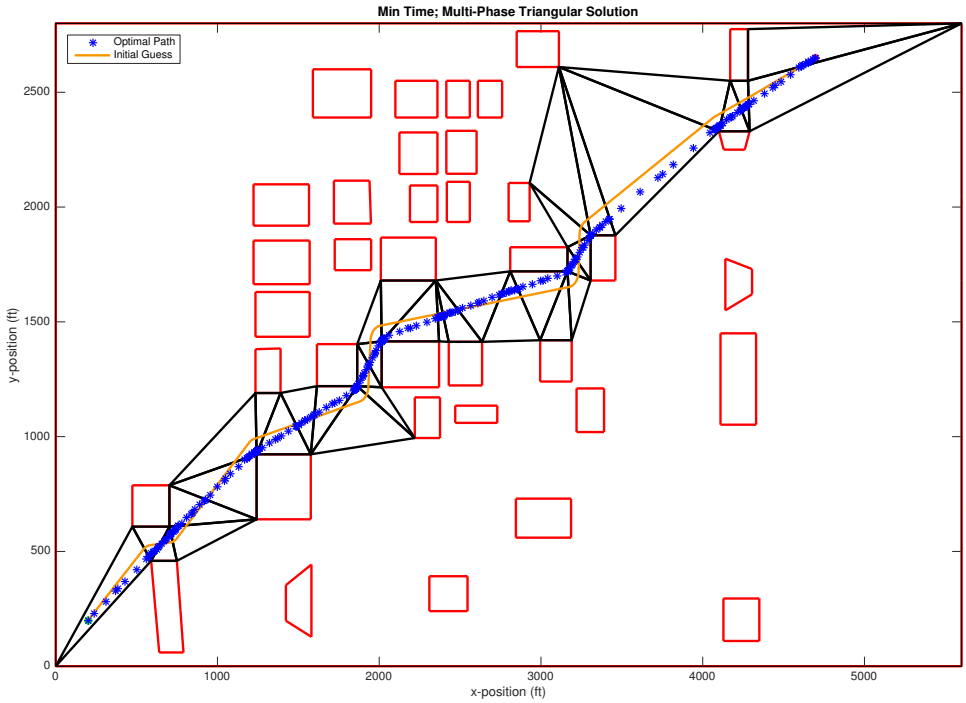

Figure 9. Hybrid Solution. Map data @2017 Google

is shown with blue asterisks. An optimal solution through the search corridor was found, however, given a quality initial guess to the path solution, the computation time for the optimal control software decreased to 3.58 seconds. The heading angle, heading angle rate, and control are shown in Figure 10 with the initial guess resulting from the Triplanner toolkit solution shown with the red dotted line and the blue solid line describing the optimal heading angle and control. Here the angle requirements between the Triplanner solution and the optimal control solution have been minimized and when compared to the mid-point solution, the Triplanner

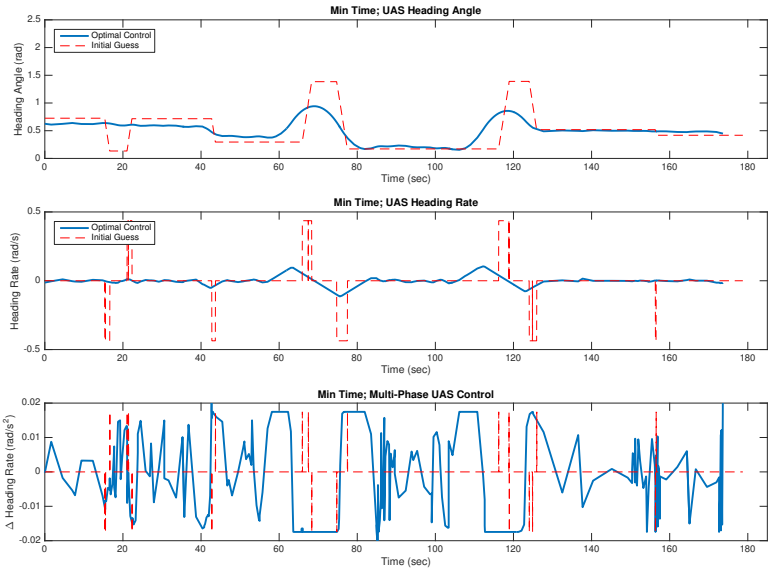

Figure 10. Hybrid Theta and Control Solution.

solution requires more control than the optimal solution but significantly less than the mid-point solution.

The two simulations shown above were each solved with an NLP tolerance of 10^{-3}. Through simulation, it was determined that decreasing the NLP tolerance to a lower threshold significantly increased the computational time while improving the objective value by just a few tenths of a second. To further decrease the computational time, the minimum flight safety buffer, defined in Equations 48 - 50, can be removed from the optimal control equality constraints. This safety buffer can instead be incorporated inside the polygon of the original building constraints resulting in equivalent objective costs while reducing the computation time to 2.23 seconds. Table 2 shows the significant differences in computation time that can be attained by incorporating the Triplanner algorithm as the initial guess to the NLP and maintaining the building safety buffer inside the polygon building constraint model.

Table 2. Table of simulation results (using SNOPT)

Initial NLP Guess	**Comp Time** (s)	**Obj Time** (s)	Building Safety Buffer
Mid-Point	4.57	173.26	NLP Equality Constraints
Triplanner	3.58	173.63	NLP Equality Constraints
Mid-Point	3.84	174.32	In-Polygon Model
Triplanner	2.12	173.69	In-Polygon Model

Without a triangulated mesh, an initial guess to the NLP for this simulation would be difficult to generate and path constraints would be challenging to model, resulting in an optimal solution that would likely return a local minimum with excessive computation times. Here it is shown the optimal control problem can be solved in 2.12 seconds with an objective cost of 173.69. This solution requires an initial guess acquired from the Triplanner toolkit solution for the safety buffer modeled inside the polygon constraint. The Triplanner solution alone solved in the fastest time at 5.84 milliseconds, with an objective cost of 181.8. Although this solution is well within the computational limits for on-baord processing, it requires an additional eight seconds of flight time to accomplish the mission and the state and control parameters are not explicitly returned. By solving the optimal control problem in GPOPS-II, the total mission time of 173.92 seconds is the fastest solution producing both the control parameter and the state values at each collocation point. Future work will include higher fidelity aircraft dynamics while incorporating exterior disturbances, both of which have proven difficult to account for with estimation techniques, thus further validating the need for optimal control software.

VI. Conclusion

The goal of this SUAS optimal control problem was to develop a flight path through a constrained environment within the computational limits for onboard, real-time flight operations. By combining optimal control methods with path planning techniques from the field of computer animation, the search space defined in the optimal control problem has been significantly reduced and path constraints have been eliminated resulting in a smaller Jacobian matrix and reducing the required gradient set required for defining an optimal solution. Computation times have been achieved within the desired limits for a matlab based approach consisting of a solution defined by a series of simplexes forming a search corridor. The optimality of the solution is dependent on this search corridor as defined with heuristic search techniques. Although there is no guarantee the globally optimal solution will be contained within the chosen corridor, the framework presented allows for a feasible solution to be attained quickly, with common input parameters, increasing the robustness of the optimal control software.

In traditional methods for optimal control, constraints must be modeled with differentiable functions such that gradient-based solvers can find optimal solutions outside of path constraints. Often, it becomes challenging to adequately represent a constrained field and the additional function evaluation result in increased computational time. By discretizing the search space with a triangulated mesh, constraints are eliminated from the search space and computation times can be greatly reduced bridging the gap between optimal control solvers and onboard, real-time solution.

Acknowledgments

The authors would like to thank the Air Force Research Laboratories, Aerospace Systems Directorate, Power and Controls Division, who sponsored the work and provided the challenging problem concept, background information, and continual support.

References

[1] *Joint Concept of Operations for Unmanned Aircraft Systems*, US Department of Defense, Washington, D.C., 2011.

[2] Heidlauf, P. T., *Optimal UAV Path Planning with Dynamic No-Fly-Zones for Target Geolocation using Line-of-Bearing Measurements and Kalman Filtering*, Ms thesis, Air Force Institute of Technology, 2017.

[3] Smith, N. E., *Optimal Collision Avoidance Trajectories for Unmanned/Remotely Piloted Aircraft*, Dissertation, Air Force Institute of Technology, 2014.

[4] Humphreys, C. J., Cobb, R. G., Jacques, D. R., and Reeger, J. A., "Dynamic Re-plan of the Loyal Wingman Optimal Control Problem in a Changing Mission Environment," *AIAA Infotech @ Aerospace*, No. January, 2017, pp. 1–13.

[5] Masternak, T. J., *Multi-Objective Trajectory Optimization of a Hypersonic Reconnaissance Vehicle with Temperature Constraints*, Dissertation, Air Force Institute of Technology, 2014.

[6] Carr, R. W. and Cobb, R., "An Energy Based Objective for Solving an Optimal Missile Evasion Problem," *AIAA Guidance, Navigation, and Control Conference*, No. 88, 2017, pp. 1–18.

[7] Carr, R. W. and Lagimoniere, E., "A Range Safety Footprint Analysis for the Dream Chaser Engineering Test Article Using Trajectory Optimization," *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013.

[8] Benson, D., Huntington, G., Thorvaldsen, T., and Rao, A., "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 6, 2006, pp. 1435–1440.

[9] Huntington, G., *Advancement and analysis of a Gauss pseudospectral transcription for optimal control problems*, Dissertation, Massachusetts Institute of Technology, 2007.

[10] Miller, J. K., Llanos, P. J., and Hintz, G. R., "Optimal Control Framework for Impulsive Missile Interception Guidance," *Guidance, Navigation, and Control Conference*, No. 8, Boston, MA, 2014, pp. 1–14.

[11] Betts, J. T., "Practical methods for optimal control and estimation using nonlinear programming," *Advances in design and control*, 2008, pp. 434.

[12] Suplisson, A. W., *Optimal Recovery Trajectories for Automatic Ground Collision Avoidance Systems (Auto GCAS)*, Dissertation, Air Force Institute of Technology, 2015.

[13] Garg, D., *Advances In Global Psuedospectral Methods For Optimal Control*, Dissertation, University of Florida, 2011.

[14] Burden, R. L. and Faires, J. D., *Numerical Analysis*, 2010.

[15] Trefethen, L. N., *Spectral Methods in Matlab*, Vol. 10, 2000.

[16] Jodeh, N. M., *Optimal UAS Assignments and Trajectories for Persistent Surveillance and Data Collection from a Wireless Sensor Network*, Dissertation, Air Force Institute of Technology, 2015.

[17] Jaklic, Ales; Leonardo, Ales; Solina, F., *Segmentation and Recovery of Superquadrics*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2013.

[18] Lewis, L. R., *Rapid motion planning and autonomous obstacle avoidance for unmanned vehicles*, Ms thesis, Naval Postgraduate School, 2006.

[19]Hurni, M., "An information-centric approach to autonomous trajectory planning utilizing optimal control techniques," 2009, pp. 296.

[20]Mohan, K., "Optimal Path Planning and Trajectory Optimization for," *University of Florida*, 2011, pp. 1–85.

[21]Zollars, M. D. and Cobb, R. G., "Simplex Methods for Optimal Control of Unmanned Aircraft Flight Trajectories," *Proceedings of the ASME 2017 Dynamics Systems and Controls Conference*, 2017, p. 10.

[22]Kapadia, M. and Badler, N. I., "Navigation and steering for autonomous virtual humans," *Wiley Interdisciplinary Reviews: Cognitive Science*, Vol. 4, No. 3, 2013, pp. 263–272.

[23]Kallmann, Marcelo; Kapadia, M., *Geometric and Discrete Path Planning for Interactive Virtual Worlds*, Morgan & Claypool Publishers series, 2016.

[24]Juarez-Perez, A. and Kallmann, M., "Full-body behavioral path planning in cluttered environments," *Proceedings of the 9th International Conference on Motion in Games - MIG '16*, 2016, pp. 107–112.

[25]Chew, P. L., "Constrained Delaunay Triangulations," *Proceedings of the third annual symposium on Computational Geometry*, Waterloo, Ontario, 1987, pp. 215–222.

[26]Demyen, D. and Buro, M., "Efficient Triangulation-Based Pathfinding," *Proceedings of the 21st National Conference on Artificial Intelligence, AAAI*, AAAI Press, Menlo Park, 2006, pp. 942–947.

[27]Dechter, Rina; Judea, P., "Generalized best-first search strategies and the optimality of A*," *Journal of the ACM*, Vol. 32, No. 3, 1985, pp. 505–536.

[28]Hetland, M. L., *Python Algorithms: Mastering Basic Algorithms in the Python Language*, Apress, 2014.

[29]Kallmann, M., "Shortest Paths with Arbitrary Clearance from Navigation Meshes," *Proceedings of the Eurographics SIGGRAPH Symposium on Computer Animation SCA*, 2010.

[30]Kallmann, M., "Dynamic and Robust Local Clearance Triangulations," *Acm Transactions on Graphics*, Vol. 33, No. 5, 2014, pp. 17.

[31]Kallmann, M. and Kapadia, M., "Navigation meshes and real-time dynamic planning for virtual worlds," *ACM SIGGRAPH 2014 Courses*, 2014, pp. 1–81.

[32]Lee, D. and Preparata, F., "Euclidean shortest paths in the presence of rectilinear barriers," *Networks 14*, 1984, pp. 393–410.

[33]Chazelle, B., "A theorem on polygon cutting with applications," *23rd IEEE Symposium on Foundations of Computer Science*, 1982, pp. 339–349.

[34]Hershberger, J. and Snoeyink, J., "Computing minimum length paths of a given homotopy class," *Computational Geometry: Theory and Applications*, Vol. 4, No. 2, 1993, pp. 63–97.

[35]Kallmann, M., "Flexible and Efficient Navigation Meshes for Virtual Worlds Flexible and Efficient Navigation Meshes," *Simulating Heterogenous Crowds iwth Interactive Behaviors*, 2016.

[36]Xu, J. and Güting, R. H., "Querying visible points in large obstructed space," *GeoInformatica*, Vol. 19, No. 3, 2015, pp. 435–461.

[37]Brass, Peter; Vigan, Ivo; Xu, N., "Shortest path planning or a tethered robot," *Computational Geometry*, Vol. 48, No. 9, 2015, pp. 732–742.

[38]Tanner, M. M., Burdick, J. W., and Nesnas, I. A. D., "Online motion planning for tethered robots in extreme terrain," *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 5557–5564.

[39]Kallmann, M., "Navigation Queries from Triangular Meshes," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 6459 LNCS, 2010, pp. 230–241.

[40]Zollars, M. D., Cobb, R. G., and Grymin, D. J., "Simplex Solutions for Optimal Control Flight Paths in Urban Environments," *Journal of Aeronautics and Aerospace Engineering*, Vol. 6, No. 3, 2017, pp. 8.

[41]Warren, J., Schaefer, S., Hirani, A. N., and Desbrun, M., "Barycentric coordinates for convex sets," *Advances in Computational Mathematics*, Vol. 27, No. 3, 2007, pp. 319–338.

[42]Schindler, M. and Chen, E., "Barycentric Coordinates in Olympiad Geometry," 2012, pp. 1–40.

[43]Visser, T., De Visser, C. C., and Van Kampen, E.-J., "Quadrotor System Identification Using the Multivariate Multiplex B-Spline," *AIAA Atmospheric Flight Mechanics Conference*, No. January, 2015, pp. 1–13.

[44]Meyer, M., Barr, A., Lee, H., and Desbrun, M., "Generalized Barycentric Coordinates on Irregular Polygons," *Journal of Graphics Tools*, Vol. 7, No. 1, 2002, pp. 13–22.

[45]Floater, M. S., "Mean value coordinates," *Computer Aided Geometric Design*, Vol. 20, No. 1, 2003, pp. 19–27.

[46]Patterson, M. A. and Rao, A. V., "GPOPS II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hpAdaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," *ACM Transactions on Mathematical Software*, Vol. 41, No. 1, 2010, pp. 1–39.

[47]Betts, J. T., "Sparse Jacobian updates in the collocation method for optimal control problems," *Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 3, 1990, pp. 409–415.

Appendix D. Journal of Aeronautics & Aerospace Engineering

Appendix D contains the third paper published in this research effort. This work explores simplex channels through narrow corridors of the urban map of Chicago, USA. Acceleration control allows for the vehicle to achieve a smaller turning radius and therefore a more direct path to the terminal position. This paper was published in the Journal of Aeronautics and Aerospace Engineering.

Journal of Aeronautics & Aerospace Engineering

Zollars et al., J Aeronaut Aerospace Eng 2017, 6:3
DOI: 10.4172/2168-9792.1000197

Research Article OMICS International

Simplex Solutions for Optimal Control Flight Paths in Urban Environments

Zollars MD[1]*, Cobb RG[1] and Grymin DJ[2]

[1]Department of Aeronautics and Astronautics, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH 45433, USA
[2]Control Science Center of Excellence, Air Vehicles Directorate, Air Force Research Laboratory, Wright-Patterson Air Force Base, OH 45433, USA

Abstract

This paper identifies feasible fight paths for Small Unmanned Aircraft Systems in a highly constrained environment. Optimal control software has long been used for vehicle path planning and has proven most successful when an adequate initial guess is presented flight to an optimal control solver. Leveragingfast geometric planning techniques, a large search space is discretized into a set of simplexes where a Dubins path solution is generated and contained in a polygonal search corridor free of path constraints. Direct optimal control methods are then used to determine the optimal flight path through the newly defined search corridor. Two scenarios are evaluated. The first is limited to heading rate control only, requiring the air vehicle to maintain constant speed. The second allows for velocity control which permits slower speeds, reducing the vehicles minimum turn radius and increasing the search domain. Results illustrate the benefits gained when including speed control to path planning algorithms by comparing trajectory and convergence times, resulting in a reliable, hybrid solution method to the SUAS constrained optimal control problem.

Keywords: Optimal control; Optimization; Unmanned systems; Direct orthogonal collocation; Path planning; Triangulated mesh

Introduction

The Department of Defense (DoD) has continued to recognize Small Unmanned Aircraft Systems (SUAS) as critical assets and the demand on their capabilities continues to grow. They are ideally suited for the dangerous or repetitive missions that otherwise require human involvement [1]. Incorporating SUAS into the battlefield will streamline systems, sensors, and analytical tasks while significantly reducing the risk to human life [2]. Across the DoD and civilian industry, the demand for unmanned capabilities has become paramount. Specifically, Manned Unmanned Teaming (MUM-T) is one role SUAS perform that augment and enhance human capabilities with a desired goal to ensure operations in complex and contested environments [3]. Manned aircraft flying through terrain and over urban canyons can experience ground threats that significantly reduce their ability to accomplish the mission. By teaming with SUAS, the manned aircraft can maintain a safe distance from the threat environment while relying on SUAS to augment the mission through system sensors. This scenario becomes ideal if the SUAS can autonomously navigate through a constrained environment from one area of interest to the next without the requirement for human interface.

Optimal control techniques are evaluated herein to determine feasible flight paths for autonomous SUAS through a highly constrained environment. Three common challenges are addressed herein that become problematic when using optimal control software. First, convergence to a solution is not always guaranteed. Second, the computation time required to achieve a solution can vary greatly. Third, constraint modeling and implementation can significantly affect the computation speed and convergence of the problem. Each of these issues can be attributed to the problem formulation, constraint implementation, and the initial guess provided to the NLP solver. Further, system parameters must be bounded appropriately to ensure the space is adequately searched, increasing the number of parameters the user is required to input.

To overcome these issues, insight will be taken from developments in the field of computer animation where Constrained Delaunay Triangulation (CDT) techniques are used to eliminate constraints from the search field and input parameters are generalized through a transformation to barycentric coordinates in a multi-phased approach. Computer animation path planning algorithms have become computationally efficient and perform effectively in moving autonomous agents through simulated environments. However, these algorithms are often restricted to the two-dimensional plane with limited control on the agent. Combining these path trajectories with the increased capabilities of optimal control software allows for efficient, feasible, multi-control solution for autonomous SUAS flight.

Background

Numerical solutions to optimal control problems are often solved with indirect or direct methods. Indirect methods use the calculus of variation to form the Hamiltonian, resulting in a two-point boundary value problem. The optimal solution is determined by solving the first-order optimality conditions while minimizing the Hamiltonian with respect to the control. With this method, a good approximation is required for the states, co-states, control and time. However, the optimality conditions can often be difficult to formulate and determining a realistic estimate of the co-states is not intuitive.

Alternatively, direct methods transcribe the infinite-dimensional optimal control problem into a finite-dimensional optimal control problem with algebraic constraints, also known as a Nonlinear Programming (NLP) problem [4]. Solutions are acquired using orthogonal collocation methods, polynomial approximation of the state, and numerical integration through Gaussian quadrature. The state, X, is approximated at a set of collocation points described as

***Corresponding author:** Zollars MD, Department of Aeronautics and Astronautics, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH 45433, USA, Tel: +1 937-255-6565; E-mail: michael.zollars@afit.edu

Received August 11, 2017; **Accepted** August 28, 2017; **Published** August 31, 2017

Citation: Zollars MD, Cobb RG, Grymin DJ (2017) Simplex Solutions for Optimal Control Flight Paths in Urban Environments. J Aeronaut Aerospace Eng 6: 197. doi: 10.4172/2168-9792.1000197

$$\tilde{x}(\tau) \approx \tilde{x}_N(\tau) = \sum_{i=1}^{n+1} x_i L_i(\tau) \quad (1)$$

where x_i represents the weight function, $L_i(\tau)$ is the Lagrange polynomial basis

$$L_i(\tau) = \prod_{\substack{i=0 \\ j\neq 0}}^{n} \frac{\tau - \tau_j}{\tau_i - \tau_j} \quad (2)$$

and τ represents an affine transformation of the time t on the interval (-1, 1)by

$$\tau = \frac{2t - (t_f + t_0)}{t_f - t_0} \quad (3)$$

This method is termed global as each collocation point is solved simultaneously rather than other fixed interval methods such as a 3 or 5 point formula method [5].

One disadvantage of the direct method results from the discretization of the optimal control problem producing several minima, leading to a solution that may be far from the optimal. To minimize this affect, an accurate prediction of the solution, control, and time are required to assure feasible results as there is no guarantee of convergence to a global minimum with direct methods. Many algorithms have been proposed previously to acquire an initial guess to the solution, including Dubins path algorithms [6] and heuristics [7,8] with computation time and accuracy being the limiting factor for complete hybrid solutions. The research herein examines the effectiveness of using computationally efficient path planning algorithms from the field of computer animation to seed the NLP used in the optimal control software for SUAS path trajectories in constrained environments.

Methodology

To properly formulate the SUAS path planning optimal control problem, all state and control variables must be defined and properly bounded and an initial guess to the path solution, control, and time must be formulated. Often, determining realistic bounds on the states, control, and time can be challenging. Bounds that are set too loose can result in high computation times while setting bounds too tightly can limit the solution search space. Further, solution accuracy and computation times are greatly dependent on the quality of the initial guess used to seed the NLP. To minimize the impacts of these issues, the optimal control problem is formulated in a phased approach. The search space is discretized into a CDT and translated into barycentric coordinates, providing standardized bounds on the system states. Path planning algorithms designed for computer animation are used to achieve feasible path solutions and are formulated to provide a quality initial guess for the states, control, and time in the optimal control problem.

Triplanner Toolkit

An extensive review of path planning through environments with clearances and algorithms developed to determine shortest paths while providing a minimum clearance to all constraints are provided by Kallmann [9,10]. These algorithms focus on computational efficiency while also providing a framework for dynamic addition and removal of constraints. They have been implemented in the 2010 version of the Triplanner toolkit[1]. An overview of the relevant algorithms from the Triplanner toolkit is given below; a more extensive review of the algorithm can be found by Kallmann, M [9,11].

[1]http://graphics.ucmerced.edu/software/Triplanner /

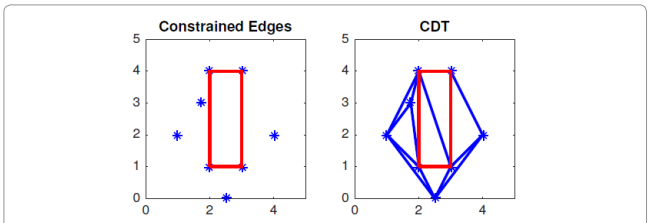

Figure 1: CDT of polygonal constraint.

First, let G consist of a planar straight-line graph with S defining a set of n segments that form all the constrained edges in the domain. A CDT, T, is then formed such that all segments of S are also segments of T and the constrained Delaunay criterion defined below are upheld.

For each unconstrained edge e of T, there exists a circle C such that

1. The endpoints of edge e are on the boundary of C
2. If any vertex v of G is in the interior of C then it cannot be "seen" from at least one of the endpoints of e [12].

Figure 1 illustrates the CDT for a single polygonal constraint.

With this technique, constraints can effectively be forced in the discretization of the space. In computer animation, these constraints represent walls, furniture, and other common obstacles an autonomous agent must avoid when traversing through a space. To account for the width of the autonomous agent, a test is performed to assure a disk of radius r can traverse through any given region without crossing a constrained edge. This allows for an efficient computation of paths of arbitrary clearance. To assure the accuracy of the feasible paths, a local clearance test is performed to verify a path solution with minimum radius of $2r$. In the event a path corridor is restricted, a refinement of the mesh is attempted by redistributing the triangulation or adding a vertex point to a straight line segment of the set S. The final triangulated mesh is then termed a "Local Clearance Triangulation (LCT)".

A path through the LCT is defined as a "free" path if it traverses from an initial point p to a final point q without crossing a constrained edge. A free path will cross several unconstrained edges resulting in a "channel" of connected simplexes formed of all traversed triangles. A path solution through this channel is determined with a "funnel" algorithm developed by Lee and Preparata, and Chazelle [13,14] as cited by Hershberger [15]. The funnel algorithm has been demonstrated under multiple applications, including path finding for autonomous agents [16], querying visible points in large data sets to define shortest paths [17], shortest paths for tethered robots [18], and robots in extreme terrain [19].

Given a corridor defined by a series of triangles, the funnel algorithm determines the shortest path from an initial point p to a final point q, subject to a defined clearance from each simplex edge. The apex of the first triangle is defined as a, with the remaining two vertex points on the shared triangle edge defined as u and v. The remaining vertex of the second triangle is defined as w. If the straight line path from a to w is feasible, that path is stored as shown in Figure 2A. Maintaining a as the apex, the straight line path from a to the following triangle vertex point, w' is evaluated for feasibility and stored if accepted, as shown in Figure 2B. This process continues until a straight line solution fails upon which the vertex providing the shortest distance to the next point in the path is chosen as the new apex, a' and the algorithm continues as shown in Figure 2C. A detailed description of the funnel algorithm can be found in Hershberger's work [15]. Finally, in order to account

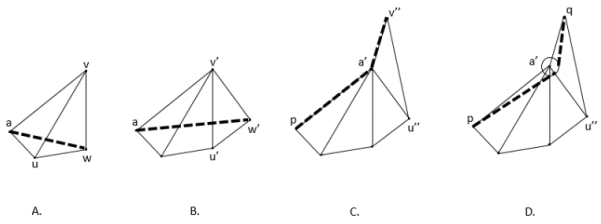

Figure 2: Triplanner path development.

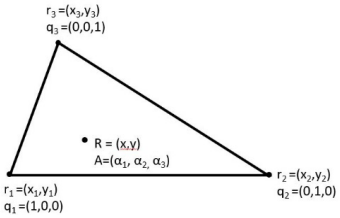

Figure 3: Barycentric coordinate frame.

for local clearances around obstacles, a circular constraint of radius r is imposed on each vertex point as illustrated in Figure 2D [11].

The Triplanner toolkit utilizes an A* search algorithm to provide a locally optimal search, defining a Dubins path solution contained in a series of triangles. It is capable of achieving path solutions on the order of milliseconds for environments with 60K+ segments [9]. This path solution can be translated to the SUAS problem by setting the radial clearance distance of each vertex equal to the turning radius of the SUAS, therefore providing a feasible path to seed the NLP. Although there is no guarantee that the defined search corridor will contain a global solution, it will guarantee a feasible flight path that is free of constraints when exogenous inputs are excluded. Currently, the Triplanner algorithm results only produce a path solution without influence of control parameters or rate limits. Although the algorithm is computationally efficient, additional work is required to produce SUAS flight trajectories while fully exploiting vehicle control parameters throughout the problem domain.

Coordinate Transformation

With a feasible path solution acquired to seed the NLP, the parameter bounds on the states, control, and time of the optimal control problem can be simplified with a translation from the Cartesian coordinate frame to the barycentric coordinate frame. Often, when dealing with simplex shapes, the barycentric coordinate frame is preferred in which the location of a point within a simplex shape is defined as a weighted measure to each of the vertices, also referred to as areal coordinates when restricted to the two-dimensional simplex [20].

Defining the coordinate system in $\mathbb{R}^2$, let r_1, r_2, and r_3 be vertices of a simplex G. Any point, R, inside simplex G can be represented in terms of the vertices of G and the barycentric weights, used as a basis as follows [21-23]:

$$\boldsymbol{R} = \sum_{j=1}^{n} \alpha_j r_j \tag{4}$$

where α represents a set of real coefficients, defining the barycentric weights whose sum equals unity and r defines the vertex points in Cartesian coordinates. Requiring the weights to be positive semi-definite ensures the point is maintained inside simplex Q,

$$\alpha_j \geq 0 \forall\, j \in [1,2,3]. \tag{5}$$

The simplex parameters illustrating Cartesian coordinates in $\boldsymbol{R}$ and barycentric coordinates in $\boldsymbol{A}$ is shown in Figure 3.

For the two-dimensional triangular relationship, transformation from a barycentric coordinate frame to a Cartesian coordinate form can be accomplished through the linear transformation

$$\boldsymbol{R} = \boldsymbol{QA} \tag{6}$$

where $\boldsymbol{R} \in \mathbb{R}^2$ defines the point location inside the simplex in Cartesian coordinates, $\boldsymbol{Q} \in \mathbb{R}^{2xn}$ defines the vertex matrix of simplex $\boldsymbol{G}$ comprised of vertex points $q_j \forall\, j \in [1,2,3]$, and $\boldsymbol{A} \in \mathbb{R}^n$ defines the barycentric weight matrix. Expanding Equation 3 and solving for the first two barycentric coordinates yields

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \boldsymbol{T}^{-1}(\boldsymbol{R} - \boldsymbol{r}_3) \tag{7}$$

where T is a 2x2 matrix comprised of the vertex points of simplex Q,

$$\boldsymbol{T} = \begin{pmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - x_3 \end{pmatrix} \tag{8}$$

and third barycentric weight, α_3, is expressed in terms of the first two calculated weights to sum to unity.

Expanding Equation 4 yields the barycentric weights in terms of both the interior point location and the vertex points of the simplex.

$$\alpha_1 = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{det(T)} \tag{9}$$

$$\alpha_2 = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{det(T)} \tag{10}$$

$$\alpha_3 = 1 - \alpha_1 - \alpha_2 \tag{11}$$

Differentiating the weights with respect to the x and y position allows for the propagation of dynamic state equations through an individual simplex.

$$\dot{\alpha}_1 = \frac{(y_2 - y_3)\dot{x} + (x_3 - x_2)\dot{y}}{det(T)} \tag{12}$$

$$\dot{\alpha}_2 = \frac{(y_3 - y_1)\dot{x} + (x_1 - x_3)\dot{y}}{det(T)} \tag{13}$$

$$\dot{\alpha}_3 = -\dot{\alpha}_1 - \dot{\alpha}_2 \tag{14}$$

Evaluating the determinant of matrix T, singularities will become problematic only if the vertex points of the simplex become collinear. By defining the discretization of the search space to hold the properties of a CDT, singularities in the dynamics will be avoided.

Optimal Control Problem Setup

The optimal control problem is formulated in the General Purpose Optimal Control Software (GPOPS-II) and implemented in MATLAB. GPOPS-II is a computation tool for solving multiple-phase optimal control problems using variable-order Gaussian quadrature collocation methods with an adaptive mesh refinement [24]. The user is required to input parameter bounds on the initial, intermediate, and final states, as well as the time vector, control, and any additional path constraints presented in the scenario.

By discretizing the problem's search space with a CDT, the path through each individual simplex can be represented in GPOPS-II as a single phase, each with a specified set of dynamics, constraints, and

bounds. The solution acquired from the Triplanner toolkit provides both the initial guess of the path solution as well as the simplex structure to effectively formulate the optimal control problem in GPOPS-II.

The output of the Triplanner algorithm yields three text files containing the path solution, the CDT, and the defined search corridor. The discretized path solution contains the endpoints of each straight line path and equally spaced points of constant radius on each turn. To properly formulate the initial guess, the solution, CDT, and the search corridor are translated to barycentric coordinates and the path is interpolated and subdivided into each simplex equating to the optimal control phases. As the path trajectory traverses across a simplex edge, the vertex points from the current phase to the next must transition such that the barycentric weights appropriately reflect the active vertex points. This process is illustrated in Figure 4.

Here it can be seen that as the path solution approaches a simplex edge, the state corresponding to the opposite vertex has no contribution to the location of the point and therefore accepts a zero value. Care must be taken to assure the state vector accurately represents the corresponding weight values as the path transitions across the simplex boundaries.

The aircraft dynamics for this problem are derived in the two-dimensional plane, representing constant altitude flight. They are formulated with a five state model describing the SUAS position in the $x(t), y(t)$ directions, the heading angle, $\theta(t)$, the heading rate, $\dot{\theta}(t)$, and the velocity *v(t)*. The control, *u(t)*, is implemented on the derivative of both the heading rate, $\ddot{\theta}(t)$, and the velocity, $\dot{v}(t)$.

$$\dot{x}^{(p)}(t) = (v)cos\left(\theta^{(p)}(t)\right) \forall\ p \in [1 \ldots P] \tag{15}$$

$$\dot{y}^{(p)}(t) = (v)sin\left(\theta^{(p)}(t)\right) \forall\ p \in [1 \ldots P] \tag{16}$$

$$\dot{\theta}^{(p)}(t) = \theta(t) \forall\ p \in [1 \ldots P] \tag{17}$$

$$\ddot{\theta}^{(p)}(t) = u_1(t) \forall\ p \in [1 \ldots P] \tag{18}$$

$$\dot{v}^{(p)}(t) = u_2(t) \forall\ p \in [1 \ldots P] \tag{19}$$

Here, *v* represents the velocity, *p* represents the current phase, and *P* defines the total number of phases in the solution, consistent with the number of simplexes in the defined search corridor.

In order to fully transform the SUAS state vector into the barycentric coordinate system, Equations 15-16 are substituted into Equations 12-13 to form the final set of dynamic equations, $\forall\ p \in [1 \ldots P]$.

$$\dot{\alpha}_1^{(p)}(t) = \frac{\left(y_2^{(p)} - y_3^{(p)}\right)(v)cos\left(\theta^{(p)}(t)\right) + \left(x_3^{(p)} - x_2^{(p)}\right)(v)sin\left(\theta^{(p)}(t)\right)}{det\left(T^{(p)}\right)} \tag{20}$$

$$\dot{\alpha}_2^{(p)}(t) = \frac{\left(y_3^{(p)} - y_1^{(p)}\right)(v)cos\left(\theta^{(p)}(t)\right) + \left(x_1^{(p)} - x_3^{(p)}\right)(v)sin\left(\theta^{(p)}(t)\right)}{det\left(T^{(p)}\right)} \tag{21}$$

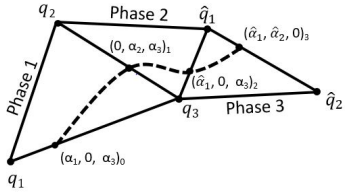

Figure 4: Simplex phased solution.

$$\dot{\alpha}_3^{(p)}(t) = -\dot{\alpha}_1^{(p)}(t) - \dot{\alpha}_2^{(p)}(t) \tag{22}$$

$$\dot{\theta}^{(p)}(t) = \ddot{\theta}(t) \tag{23}$$

$$\ddot{\theta}^{(p)}(t) = \psi(t) \tag{24}$$

$$\dot{v}^{(p)}(t) = a(t) \tag{25}$$

The control is implemented on the derivative of the velocity and heading rate,

$$\boldsymbol{u}^{(\boldsymbol{p})}(\boldsymbol{t}) = \left[\boldsymbol{\psi}(\boldsymbol{t})\ \boldsymbol{a}(\boldsymbol{t})\right] \tag{26}$$

The state vector is defined with six states, represented as

$$\boldsymbol{X} = \left[\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_3, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{v}\right] \tag{27}$$

Subject to these dynamic constraints, the objective for each scenario herein is to minimize the cost functional

$$J^{(p)} = \int_{t_i^{(p)}}^{t_f^{(p)}} dt \tag{28}$$

$$J = \sum J^{(p)} \tag{29}$$

given the initial and final boundary constraints describe as

$$\boldsymbol{X}^{(1)}\left(t_0^{(1)}\right) = \left[(\alpha_1)_0, (\alpha_2)_0, (\alpha_3)_0\right] \tag{30}$$

$$\boldsymbol{X}^{(\boldsymbol{P})}\left(t_f^{(P)}\right) = \left[(\alpha_1)_f, (\alpha_2)_f, (\alpha_3)_f\right] \tag{31}$$

where the heading, heading rate, and velocity are free variables in the initial and final state. Further, inequality constraints are implemented to maintain the search space within each simplex and provide bounds to the state, control and time defined as

$$0 \le \alpha_1^{(p)} \le 1 \tag{32}$$

$$0 \le \alpha_3^{(p)} \le 1 \tag{33}$$

$$0 \le \alpha_3^{(p)} \le 1 \tag{34}$$

$$\left|\theta^{(p)}\right| \le \pi \tag{35}$$

$$\left|\dot{\theta}^{(p)}\right| \le 25\ deg/s \tag{36}$$

$$\left|u_1^{(p)}\right| \le 1\ deg/s^2 \tag{37}$$

$$\left|u_2^{(p)}\right| \le 2\ ft/s^2 \tag{38}$$

$$0 \le t^{(p)} \le \frac{l_{max}^{(p)}}{v} \tag{39}$$

where l_{max} describes the longest edge of the current simplex. The bound on the fourth and fifth state were chosen to represent a general group 1 SUAS [1]. The bound on the heading rate control was chosen such that the $\dot{\theta}$ vector represented an appropriate set of dynamics to implement in an aircraft control system.

Finally, event constraints are implemented to assure a continuously smooth transition of the state variables as the path traverses through each phase, described as

$$X_0^{(p)} - X_f^{(p-1)} = 0 \forall\ p \in [2 \ldots P]. \tag{40}$$

Scenarios

Two scenarios were evaluated to illustrate the savings in the objective cost when solving for constrained path trajectories with the optimal control software, GPOPS-II. In each scenario presented, all polygon constraints are convex, however the approach can be applied to arbitrary polygons. The first scenario considered an aircraft flying at max speed with control limited to only the change in heading rate. This reduces the previously defined state matrix in Equation 27 to a five state model defined as

$$\boldsymbol{X} = \left[\lambda_1 \lambda_2 \lambda_3 \theta \dot{\theta}\right] \tag{41}$$

while the control, previously defined in Equation 26, is reduced to

$$\boldsymbol{u}^{(p)}(t) = \left[\psi(t)\right]. \tag{42}$$

The Triplanner solution was determined with a maximum radial off-set distance defined by the vehicles bank angle limit when flying at max speed. The path results, along with the CDT discretization, were used as inputs to seed the NLP of the optimal control software.

The second scenario is constructed to illustrate the advantages of path planning when allowing for speed control on an air vehicle. Again, the Triplanner algorithm is used to determine an initial path solution and CDT discretization. In contrast to the first scenario, the radial off-set distance is now defined using the minimum allowable air speed of the SUAS. This reduces the minimum turn radius and may increase the feasible search space of the problem. The optimal control problem consists of the six state, two control model defined previously.

For both scenarios, the SUAS is required to fly through a pre-defined area of downtown Chicago, USA, measuring *5600 x 2800 ft*. The altitude of the SUAS is restricted to *600 ft* AGL and therefore all structures exceeding a height of *550 ft* are modeled as path constraints that must be avoided. The initial and final locations of the path are defined as

$$(x_i, y_i) = (200, 200) \tag{43}$$

$$(x_f, y_f) = (2630, 2650). \tag{44}$$

The final location of the scenario was chosen such that the most direct path would require the SUAS to navigate through narrow building corridors requiring minimum radius turns thus illustrating the search domain of the problem.

The initial guess of the path trajectory supplied to the NLP solver is acquired through the Triplanner algorithm as described previously. The initial guess of the heading vector is determined by the angle between consecutive Cartesian coordinates of the Triplanner solution. The heading rate and control are calculated with a right point finite differencing method initiated with the heading angle vector. Each of these vectors are rate limited to remain consistent with those used in the optimal control problem as in Equations 36 and 37. The initial guess for the velocity vector is formulated with maximum speed on straight sections of the path and minimum speed on the minimum radius turns while the acceleration vector is initiated with the zero vector. The time vector is approximated through each phase as the running summation of the Euclidean distance between consecutive points divided by the vehicle airspeed.

The constraint map is shown in Figure 5 with each building exceeding *550 ft* described with a red enclosed polygon. Building heights were estimated in order to construct a formidable optimal control problem. The initial and final path locations are shown with green and red asterisks respectively.

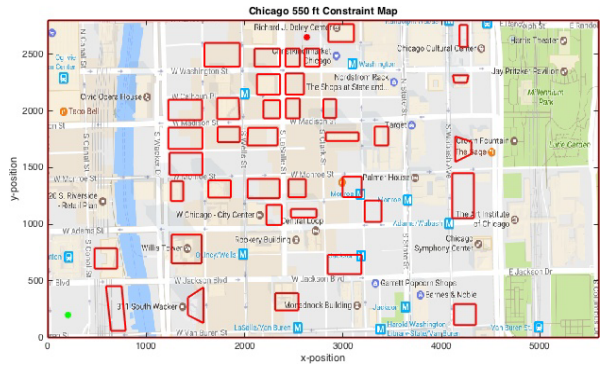

Figure 5: Chicago constraint map. Map Data @2017 Google.

GPOPS-II User Settings	
Mesh Method	hp-Patterson Rao
Mesh Tolerance	10^{-2}
NLP Solver	SNOPT
Method	RPM-differential
Derivative Supplier	AdiGator
Derivative Level	First
NLP Tolerance	10^{-3}
Min Collocation Points	4
Max Collocation Points	10
Mesh Fraction	0.5* ones (1, 2)
Mesh Collocation Points	4* ones (1, 2)

Table 1: GPOPS user defined settings.

The GPOPS-II user settings defined for each scenario are described as shown in Table 1.

Minimum Time Scenario with Max Speed

The optimal control problem for the first scenario is as described previously with the objective being to fly from the initial point to the final point in the shortest amount of time. Often, with minimum time SUAS problems, the path solution is flown at maximum speed, therefore this problem only allows a single control defined as the change of heading rate of the vehicle.

Scenario #1: Triplanner solution

The Triplanner algorithm is solved and implemented as the initial guess to the NLP. It is initiated with the polygonal constraints, the initial and final location of the path solution, and a defined off-set distance from each constraint. To assure a feasible flight path solution, the radial off-set distance is determined through the relationship between the vehicles velocity and turn rate as follows,

$$R = \frac{v}{\omega} \tag{45}$$

for R is the minimum turn radius, v is the velocity, and ω is the turn rate. For this scenario, the max velocity was set to *30 ft/s* with a turn rate of *25 deg/s* yielding a turn radius of *68 ft*. The resulting search corridor and path solution are shown in Figure 6.

The Triplanner solution is solved in *4.07* milliseconds on a PC resulting with an objective time of *134* seconds. Here the constraint

off-set distance is shown on each polygonal vertex with blue circles. Due to the narrow corridors defined between buildings, and the maximum required off-set distance, the only feasible path solution requires the SUAS to fly around the constraints as shown in the black outlined simplex search corridor. The path solution is shown as a Dubins path made of up straight line sections and max radius turns. However, this path is not optimal due to the placement of the circular off-set constraints placed on the vertex of each polygonal constraint. This allows for improvement to be seen in the objective function when solved with an NLP.

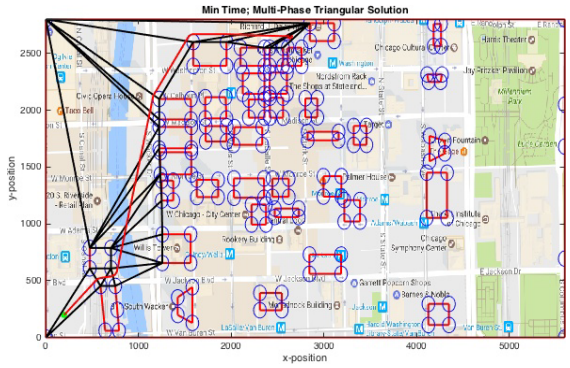

Figure 6: Max radius Triplanner solution. Map Data @2017 Google."

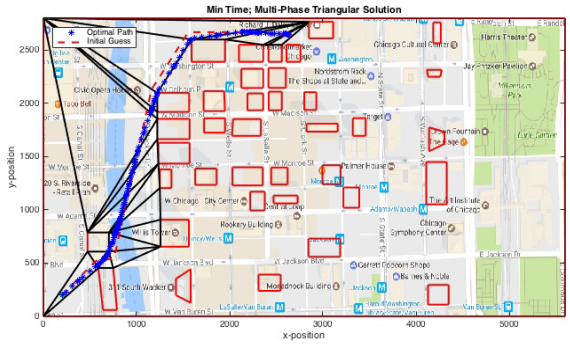

Figure 7: Sim #1 GPOPS-II Solution. Map Data @2017 Google.

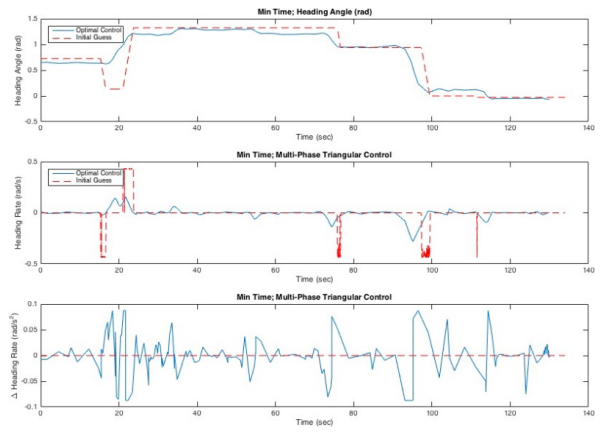

Figure 8: Sim #1 GPOPS-II states and control.

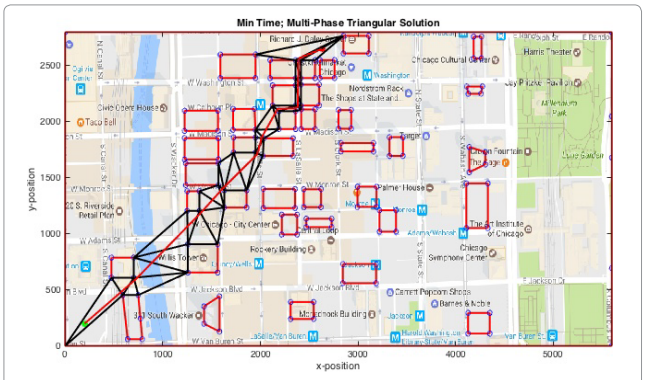

Figure 9: Min radius Triplanner solution. Map Data @2017 Google.

Scenario #1: GPOPS-II solution

The path result for the optimal solution through the defined search corridor is shown in Figure 7.

The optimal solution is solved in *2.12* seconds with an objective of *129.9* seconds. The Triplanner solution used to seed the NLP solver, SNOPT, is shown with the red dashed line while the discretized optimal solution is shown with the blue asterisks. A small improvement in the objective is seen over the Triplanner results but at the cost of computation time.

Figure 8 describes the heading, heading rate, and control respectively. The initial guess formulated from the Triplanner results can be seen with the red lines while the optimal solution is shown in blue.

Here, the difference in the two solutions is shown as the Dubins Triplanner solution requires max radius turns at each vertex along the path while the optimal control solution can blend the solution through the constrained field.

Although there are benefits to the optimal control solution, justification for using the optimal control software cannot be made at this point given the computation time required to achieve a solution with only minimal improvement to the objective.

Minimum Time Scenario with Speed Control

The optimal control problem for the second scenario consists of the six state, two control model as described previously in Equations 20-39.

Scenario #2: Triplanner solution

The Triplanner solution is again initiated with the polygonal constraints, the initial and final location of the path solution, and a defined off-set distance from each constraint. With the velocity now being a state, the SUAS has the ability to reduce speed in order to achieve a smaller turn radius and therefore navigate through narrow city corridors. However, within the constraints of the 2010 Triplanner toolkit, the turn radius cannot be varied during a simulation. This limits the Triplanner algorithm to solve for a solution using the minimum speed turn radius calculated from Equation 40, yielding a minimum turn radius of *22.9 ft* at the SUAS speed of *10 ft/s*. The Triplanner results are shown below in Figure 9.

Similar to the first scenario, the Triplanner solution resulted in just *6.1* milliseconds, but at an objective time of *364* seconds which is significantly increased due to the minimum speed restriction. Again the constraint off-set distance is shown with blue circles around each

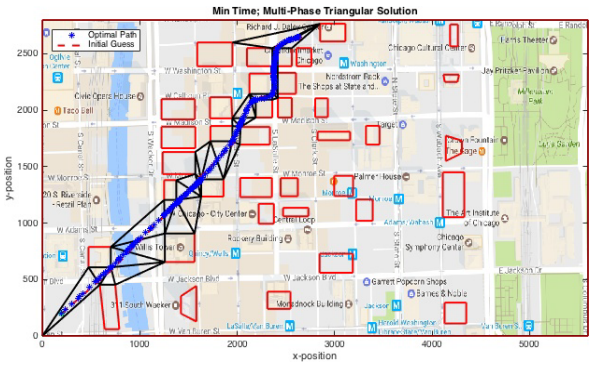

Figure 10: Sim #2 GPOPS-II solution. Map Data @2017 Google.

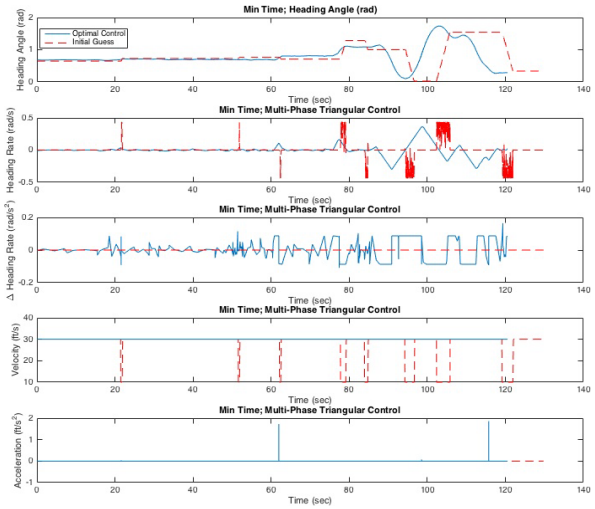

Figure 11: Sim #2 GPOPS-II states and control.

Control	Solution Method	NLP Seed	Objective Time (sec)	Computation Time
ψ	Triplanner	N/A	134	4.07 ms
ψ	GPOPS-II	Triplanner	129.9	2.12 s
a, ψ	Triplanner	N/A	364	10.4 ms
a, ψ	GPOPS-II	Triplanner	120.4	2.9 s

Table 2: Simulation results.

vertex of the search corridor and the path solution is shown with the solid red line. Under minimum speed, the search corridor provides a feasible search space that is a more direct route to the finish location. Although the distance traveled is significantly decreased, the objective time for the Triplanner solution is too long to consider this a viable solution in itself.

Scenario #2: GPOPS-II solution

The Triplanner solution will again be used as the initial guess to seed the NLP solver SNOPT. Due to the increased objective time of the minimum turn radius Triplanner solution, the input vectors are scaled in time to represent a maximum speed solution during straight sections of the path and a minimum speed solution during the constant radius turns.

Implementing the optimal control problem as described previously, but now incorporating control on the SUAS acceleration, the optimal solution through the defined search corridor is shown in Figure 10.

The optimal solution is solved in *2.86* seconds with an objective of *120.4* seconds. Here the Triplanner solution, post processed for speed control, is shown with the red dashed line while the optimal solution is shown with the blue asterisk. The computation times are similar to those found in the first GPOPS-II simulation, however, by allowing control on the SUAS speed, objective times can be significantly reduced, allowing the vehicle to traverse a more direct path to the target location.

Figure 11 describes the heading, heading rate, heading rate control, velocity, and acceleration control respectively. The initial guess formulated from the Triplanner results can be seen with the red lines while the optimal solution is shown in blue.

Similar to the first solution, the Dubins path solution resulting from Triplanner can be seen in the top subfigure but here it is acquired with minimum radius turns. By formulating the problem with optimal control software, the turn points in the path can be optimized through the constraints. Further, the 4th subfigure shows the velocity is maintained at max speed for the optimal solution, thus providing a feasible path solution that is direct to the target location and flown at maximum speed. Table 1 summarizes the simulation results.

Conclusions

This work demonstrated a solution technique to solve feasible path solutions for SUAS through a highly constrained environment. Leveraging computationally efficient algorithms developed for computer animation, a CDT was performed on the search space and a Dubins path solution was determined through a simplex search corridor, free of all path constraints. The defined search corridor, dependent on the user supplied radius off-set distance set in the Triplanner algorithm, defines the domain of the optimal control solutions space. By initiating Triplanner with a SUAS maximum speed turn radius, path results are restricted to wide simplex corridors, excluding many routes on the interior of the domain. Although these solutions are flown at maximum speed, the path is often highly sub-optimal. On the contrary, by initiating the Triplanner algorithm with the SUAS minimum speed, the defined off-set radius is reduced and path corridors through the interior of the city are included in the solution space. These solutions provide more direct routes to the final location, however, the flight time required to accomplish the path is excessive at minimum speeds.

Optimal control software is utilized to blend the two Triplanner results by allowing for control on the SUAS acceleration, enabling the aircraft to optimize the speed profile while determining a path solution through a more direct route on the interior of the city. Using the minimum SUAS turn radius to initiate the Triplanner algorithm, a Dubins path solution is acquired and used as the initial guess for the NLP. This result alone is sub-optimal as the Triplanner algorithm places the minimum turn radius path constraints on each vertex of the search corridor, defining the Dubins path. The optimal control software is able to improve on the Triplanner solution by flying a more direct path while maintaining maximum flight speed, improving the objective function by over *8%* on the most direct route.

Speed control in previous path planning algorithms for minimum time objectives, are often not included due to the complexities inherent to the design. This effort has demonstrated the benefit speed control can have in determining efficient flight trajectories in highly constrained domains. Ultimately, by including acceleration control on the SUAS,

computational efficiencies and trajectory solutions, provided by the Triplanner algorithm, can be exploited with optimal control software to produce accurate and efficient path results in minimum time.

Acknowledgments

The authors would like to thank the Air Force Research Laboratory, Aerospace Systems Directorate, Power and Controls Division, who sponsored the work and provided the challenging problem concept, background, information, and continual support.

References

1. Airspace Itegration Integrated Product Team (2011) Unmanned aircraft system airspace integration plan. Washington, DC US Department of Defense.
2. (2016) Deputy chief of staff for intelligence, surveillance, and reconnaissance (ISR) x small UAS Flight Plan 2016-2036.
3. Air Force Research Laboratory (2013) Air force research laboratory autonomy science and technology strategy.
4. Huntington G (2007) Advancement and analysis of a guess pseudospectral transcription for optimal control problems. Massachusetts Institute of Technology, USA.
5. Trefethen L (2000) Spectral Methods in Matlab. Lloydia Cincinnati.
6. Zollars M, Blue P, Brian B (2007)Wind corrected flight path planning for autonomous micro air vehicles utilizing optimzation techniques. AIAA Atmospheric Flight Mechanics Conference and Exhibit. Hilton Head, SC.
7. Ferguson D, Likhachev M, Stentz A (2005) A guide to hueristic-based path planning. Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS): 1-10.
8. Humphreys C, Cobb R, Jacques D, Reeger J (2016) A hybrid optimization technique applied to the intermediate-target optimal control problem. Global Journal of Technology and Optimization 1: 2.
9. Kallmann M (2010) Shortest paths with arbitrary clearance from navigation meshes. Proceedings of the Eurographics SIGGRAPH Symposium on Computer Animation SCA. Madrid, Spain.
10. Kallmann M (2014) Dynamic and robust local clearance triangulation. Acm Transactions on Graphics 33:17.
11. Kallmann M (2010) Naviation queries from triangular meshes. Proc. Third International Confrerence Motion in Games 2: 230-241.
12. Chew P (1987) Constrained delaunay triangulations. Proceedings of the third annual symposium on Computational Geometry: 215-222.
13. Lee D, Preparata F (1984) Euclidean shortest paths in the presence of rectilinear barriers. Networks 14: 393-410.
14. Chazelle B (1982) A theorem on polygon cutting with applications. 23rd IEEE Symposium on Foundations of Computer Science: 339-349.
15. Hershberger J, Snoeyink J (1993) Computing minimum length paths of a given homotopy class. Computational Geometry: Theory and Applications 4: 63-97.
16. Kallmann M (2016) Flexible and efficient navigation meshes for virtual worlds flexible and efficient navigation meshes. In Simulating Heterogenous Crowds iwth Interactive Behaviors.
17. Xu J, Guting R (2015) Querying visible points in large obstructed space. GeoInformatica 19: 435-461.
18. Brass P, Vigan I, Xu N (2015) Shortest path planning or a tethered robot. Computational Geometry 48: 732-742.
19. Tanner M, Burdick J, Nesnas I (2013) Online motion planning for tethered robots in extreme terrain. Proceedings - IEEE International Conference on Robotics and Automation: 5557-5564.
20. Warren J, Schaefer S, Hirani A, Desbrun M (2007) Barycentric coordinates for convex sets. Advances in Computational Mathematics 27: 319-338.
21. Visser T, De Visser C, Van Kampen E (2015) Quadrotor system identification using the multivariate multiplex b-spline. AIAA Atmospheric Flight Mechanics Conference: 1-13.
22. Meyer M, Lee H, Barr A, Desbrun M (2002) Generalized barycentric coordinates on irregular polygons. J Graphics Tools 7: 13-22.
23. Floater M (2003) Mean value coordinates. Computer Aided Geometric Design 20: 19-27.
24. Patterson M, Rao A (2015) GPOPS-II manual: A general-purpose MATLAB software for solving multiple-phase optimal control problems version. p. 1-72.

Citation: Zollars MD, Cobb RG, Grymin DJ (2017) Simplex Solutions for Optimal Control Flight Paths in Urban Environments. J Aeronaut Aerospace Eng 6: 197. doi: 10.4172/2168-9792.1000197

Appendix E. Aerospace Conference 2018

Appendix E contains the fourth paper published in this research effort. The work explores the ability to connect multiple waypoints within the simplex structure of an optimal control problem for path planning. This paper was published and presented at the IEEE 2018 Aerospace Conference, Big Sky, Montana, in March 2018.

Optimal Path Planning for SUAS Waypoint Following in Urban Environments

Michael D. Zollars & Richard G. Cobb
Department of Aeronautics and Astronautics
Air Force Institute of Technology
Wright-Patterson AFB, OH, 45433
michael.zollars@afit.edu; richard.cobb@afit.edu

David J. Grymin
Controls Science System Center of Excellence
Wright-Patterson AFB, OH, 45433
david.grymin.1@us.af.mil

Abstract*—In this research, Small Unmanned Aircraft Systems (SUAS) are used to determine feasible flight paths to multiple waypoints within an urban environment. Direct orthogonal collocation methods are used while leveraging navigation mesh techniques developed for fast geometric path planning solutions. Waypoints are included throughout a constrained city map to illustrate feasible flight paths through tightly constrained path corridors. The two-dimensional solution is achieved with a multi-phase approach defined through a discretized simplex mesh with aircraft control on acceleration and the change in the vehicle's heading rate. Constrained optimal control problems for SUAS have long suffered from excessive computation times caused by a combination of constraint modeling techniques and the quality of the initial path solution provided to the optimal control solver, ultimately preventing implementation into real-time, on-board systems. These issues are addressed herein with a new approach by triangulating the search space to define a polygonal search corridor free of constraints while alleviating the dependency of problem specific parameters by translating the problem to barycentric coordinates. Utilizing algorithms developed for geometric path planning, the initial path solution is comprised of four path intervals connected at each waypoint. These path intervals provide a constraint free search corridor defining a search domain for the optimal control software. Results are applied to illustrate two-dimensional flight trajectories through downtown Chicago at an altitude of 550 feet Above Ground Level (AGL). Computation and objective times are reported to illustrate the design implications for real-time optimal control systems.

TABLE OF CONTENTS

1. INTRODUCTION 1
2. PATH PLANNING 1
3. METHODOLOGY 2
4. ALGORITHM DEVELOPMENT 3
5. SCENARIO 5
6. RESULTS 5
7. CONCLUSIONS 6
ACKNOWLEDGMENTS 7
REFERENCES 7
BIOGRAPHY 7

1. INTRODUCTION

Path planning algorithms have become essential to the development of autonomous navigation across a wide range of applications. In agricultural development, air and ground vehicles are used to autonomously collect sensor data to be used in the prediction of crop growth [1], underwater vehicles are used to explore resources or landmarks below the surface of the water [2], and the Department of Defense uses autonomous vehicles in a multitude of capacities ranging from transports to target recognition [3]. Each of these applications can be formulated into an optimal control problem to minimize the objective of a desired cost. When including path constraints and additional waypoints into the problem formulation, another layer of realism is added but also increases the complexity of the algorithm.

Often, the path planning problem requires intermediate waypoints to be visited by the vehicle before arriving at the final destination. Many times this results in the traveling salesman approach where a cost function is developed to determine the appropriate order and frequency the waypoints should be visited. Previous work has shown optimized paths constructed to maximize the collection of data from wireless sensor nodes [4] as well as paths designed through energy fields to maximize the vehicle endurance [5]. These problems are often focused on optimizing the waypoints visited rather than the path to the waypoints or the deconfliction of a large number of path constraints.

The work herein does not evaluate the order or frequency a waypoint is visited, rather determines the optimal minimum time path between a series of selected waypoints through a highly constrained field. By discretizating the problem domain into a triangulated mesh, polygonal constraints are eliminated from the search space and the optimal control problem is reduced to a search channel free of constraints. Translating the problem to barycentric coordinates reduces the problem specific parameter bounds and thus minimizes the required inputs in the problem formulation. Previous work has shown the benefits of using a triangulated mesh over traditional formulations of the constrained optimal control problem [6] and has illustrated the technique in a realistic constrained field [7],[8].

2. PATH PLANNING

Optimal Control

Direct methods for optimal control path planning transcribe the continuous time optimal control problem into a finite dimensional problem, also known as a Nonlinear Programming (NLP) problem [9]. Algebraic constraints are formulated to limit the search space and provide bounds to the state and control vectors. The solution is acquired using orthogonal collocation methods, defining the the state vector with an orthogonal polynomial while numerical integration and differentiation are calculated through Guassian quadrature. This method is termed a global method as each collocation point is solved simultaneously with zero error and a solution can be achieved with a minimal number of collocation points. Results may however be suboptimal, as several local minima can exists in nonlinear, non-convex problems.

To increase the opportunity for an optimal control solver to achieve a global optimal solution, an accurate prediction of the state, control, and time should be used as an initial guess to the solver. Determining an accurate guess that can be formulated with computation efficiency can be challenging. Past work has utilized Dubins paths models [10], Particle Swarm Optimizations (PSO) [11], and boot strap methods within the solver [12] to seed the optimization routine. Each of these methods provide advantages and disadvantages, but they all become exponentially more complex as the number of path constraints are increased within the search domain. The work herein will use geometric path planning algorithms as an initial guess to the optimal control solver. These algorithms have been successful in eliminating polygonal path constraints from the search space by discretizing the space with a triangulated mesh and forming a polygonal search channel containing the approximated solution.

Fast Geometric Path Planning

When considering path planning for video game virtual worlds, autonomous agents are required to traverse through simulated environments while avoiding all obstacles. The final path result must be computationally efficient while taking into account path length, time, and energy expended to produce a realistic simulation operated in a real-time environment [13]. The resulting geometric path solution is determined with consideration for an autonomous agents width and body movements such that constrained edges are not violated. However, when comparing this method to path planning for Small Unmanned Aircraft Systems (SUAS), control rates and limits are not considered, providing a necessity for optimal control software.

An extensive review of fast geometric path planning with clearances is given in [14]. Algorithms to determine shortest paths while providing a minimum clearance to all constraints are provided in [15]. These algorithms focus on providing reliable, computationally efficient path solutions and provide a framework for dynamically removing or adding constraint regions within a triangulated mesh formed from a Constrained Delaunay Triangulation (CDT). The work herein implements these algorithms in the 2010 version of the Triplanner toolkit[2]. A more extensive review of the algorithm can be found in [14], [16].

3. METHODOLOGY

To properly formulate the SUAS path panning optimal control problem, all state and control variables must be defined and properly bounded and an initial guess to the path solution, control, and time are required. The Triplanner algorithm is capable of solving geometric path solution while avoiding $60K+$ constraints with computational times on the order of milliseconds [14]. This provides a fast geometric solution to be used as the initial guess to the optimal control solver. In order to most efficiently implement the Triplanner solution as the initial guess, a transformation to barycentric coordinates is required and a phased approached is formulated within the construct of the CDT.

[2]http://graphics.ucmerced.edu/software/tripath/

Coordinate Transformation

Given a set of barycentric weights [17], α_i, the Cartesian coordinates can be represented as

$$\mathbf{R} = \sum_{\mathbf{i=1}}^{\mathbf{3}} \alpha_{\mathbf{i}} \mathbf{r_i} \tag{1}$$

where $\mathbf{R}$ defines the location inside the simplex and $\mathbf{r_i}$ describes each vertex point. Expanding Equation 1 and representing the final weight α_3 in terms of the first two weights, the expression below represents the weights of the barycentric coordinate frame in terms of the Cartesian coordinates,

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = T^{-1}(\mathbf{R} - \mathbf{r_3}) \tag{2}$$

$$\alpha_3 = 1 - \alpha_1 - \alpha_2 \tag{3}$$

where T is a 2×2 matrix representing the vertices of the triangle as

$$T = \begin{pmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{pmatrix}, \tag{4}$$

and $\mathbf{R} - \mathbf{r_3}$ is a 2×1 vector summation of the Cartesian coordinates. Expanding Equation 2 gives the barycentric weights in terms of the vertex coordinates and the point location within the triangle.

$$\alpha_1 = \frac{(y_2-y_3)(x-x_3)+(x_3-x_2)(y-y_3)}{det(T)} \tag{5}$$

$$\alpha_2 = \frac{(y_3-y_1)(x-x_3)+(x_1-x_3)(y-y_3)}{det(T)} \tag{6}$$

$$\alpha_3 = 1 - \alpha_1 - \alpha_2 \tag{7}$$

Vehicle Dynamics

Within the construct of a triangulated mesh, the optimal control problem is formulated into a phased approach. Each simplex that is traversed is represented as a single phase p, and each phase is connected through event constraints equating the states, control and time at each phase boundary. The total number of phases is represented as P. The simplex corridor is defined through the results of the Triplanner algorithm and account for phases $1...P$. The SUAS is described with two-dimensional dynamics in a three-state model defined as

$$\dot{x} = (v)cos(\theta) \tag{8}$$

$$\dot{y} = (v)sin(\theta) \tag{9}$$

$$\dot{\theta} = \theta. \tag{10}$$

Transforming the coordinates into a barycentric coordinate frame and taking the derivative of the weight equations with respect to the x and y coordinates, a new set of dynamic equations can be defined to propagate the state variables through each simplex. This results in an equivalent set of dynamics, previously described in Equations 8 - 10, now defined through barycentric weights as

$$\dot{\alpha}_1^{(p)} = \frac{(y_2-y_3)(v)cos\theta+(x_3-x_2)(v)sin\theta}{det(T)} \tag{11}$$

$$\dot{\alpha}_2^{(p)} = \frac{(y_3-y_1)(v)cos\theta+(x_1-x_3)(v)sin\theta}{det(T)} \tag{12}$$

$$\dot{\alpha}_3^{(p)} = -\dot{\alpha}_1^{(p)} - \dot{\alpha}_2^{(p)} \tag{13}$$

$$\dot{\theta}^{(p)}(t) = \theta(t) \tag{14}$$

$\forall p \in [1...P]$. With the dynamics described in terms of a weighted measure to each simplex vertex, a phased solution can be constructed.

Waypoint Intervals

When the SUAS is required to visit multiple waypoints, Triplanner can be used to generate an initial guess for each interval, defined as the path between consecutive waypoints. Within each interval, a phased approach is used where each simplex represents an individual phase. As a new interval is introduced, the first phase will be identical to the last phase of the previous interval. This process is shown below in Figure 1 with δ_i defining the current interval.

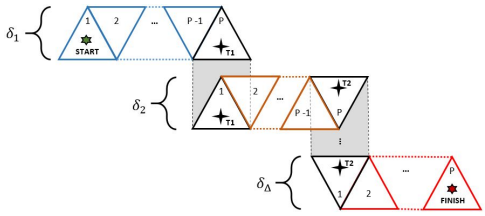

Figure 1. **Waypoint Intervals**

As the path solution terminates at a waypoint, represented at phase P in interval δ_i, the following interval begins its path at the same waypoint in the same phase, now designated as phase 1 in interval δ_{i+1}. The overlap of this phase is essential to assure a continuous transition of the optimal control problem where the waypoint locations are enforced through initial and final state constraints. This process is continued for all intervals with Δ representing the total number of intervals in the problem solution.

Optimal Control Solver

The optimal control problem presented is solved in the General Purpose Optimal Control Software (GPOPS-II) [18]. The key set-up parameters for the solver are listed in Table 1.

Table 1. GPOPS-II User Settings

GPOPS-II User Settings	
Mesh Method	hp-PattersonRao
Mesh Tolerance	$\frac{1}{2} * 10^{-2}$
NLP Solver	SNOPT
Derivative Supplier	Adigator
Method	RPM-differential
NLP Tolerance	10^{-5}
Min Collocation Points	4
Max Collocation Points	10
Mesh Fraction	$\frac{1}{4}$*ones(1,4)
Mesh Collocation Points	4*ones(1,4)

4. ALGORITHM DEVELOPMENT

The flow chart in Figure 2 describes the series of algorithms required to achieve a solution to the multiple waypoint optimal control problem. This flow chart illustrates the hybrid method, combining the fast geometric path solutions attained by Triplanner with direct orthogonal collocation methods for acquiring optimal path solutions.

Geometric Planner

The *Waypoints* algorithm described in the flow chart is initiated with the function *Define_interval*. Here, the number of waypoints in the path solution is defined along with a matrix consisting of each waypoint location. The first and last waypoints of the matrix consists of the starting location

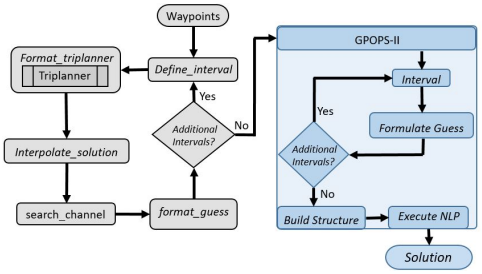

Figure 2. **Algorithm Flow Chart**

of the SUAS, thus requiring the vehicle to return to its original location. An interval is defined as the path between each waypoint location resulting in $N + 1$ intervals for N waypoints.

The *Format triplanner* algorithm selects the start and end point for the current interval and initiates the Triplanner algorithm. The output of Triplanner provides text files consisting of the constrained and unconstrained edges in the simplex mesh forming the CDT as well as the Dubins path solution. Figure 3 shows the resulting geometric path consisting of constant radius turns connected by straight line segments. The blue asterisks illustrate the results of the Triplanner solution. Constrained edges are illustrated with solid red lines, while unconstrained edges are solid gray lines. The path solution starts at the lower left green asterisk and ends at the upper right red asterisk.

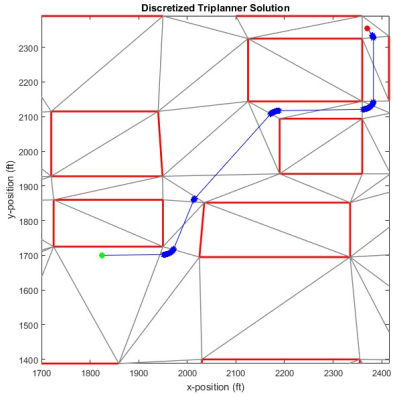

Figure 3. **Triplanner Solution**

With the resulting path from Triplanner, points are interpolated along the straight line sections of the path solution. This is accomplished based on a predefined spacing in the *interpolate solution* function. This provides a foundation for dividing the path solution into each represented simplex and constructing a properly formatted initial guess for the optimal control solver.

The function *search channel* performs three tasks. First, the interpolated path is transformed to barycentric coordinates and the path is segmented into each simplex. Second, a connectivity matrix is formed defining the order of simplexes that contain the path solution. Here, the vertex points of each simplex are arranged such that the common edge between consecutive simplexes can be identified. Finally, the connectivity matrix is augmented with three columns defining the shared simplex edge along the search corridor, the determinant of matrix T found in Equation 4, and the time required to traverse the longest edge of each simplex. Figure

4 shows the interpolated path illustrated with blue asterisks and the search channel shown with black solid lines.

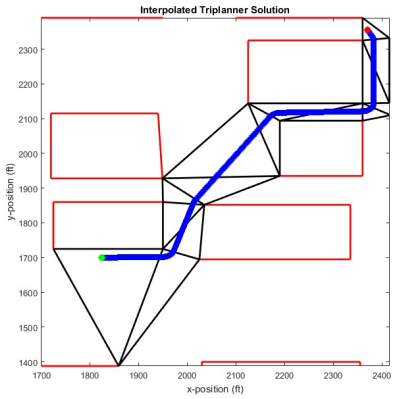

Figure 4. Interpolated Path Solution

The final function block before entering the optimal solver, *formate guess*, builds the required vectors for an appropriate guess for the optimal control problem. The heading is formulated using a three-point finite differencing scheme beginning with the position vector of the interpolated path solution. Since the Triplanner results in a Dubins path, the heading rate and change in heading rate are defined by the SUAS rate limits based on the heading angle. The velocity vector is formed with a maximum speed during straight line portions of the path and minimum speed during the constant radius turns. This allows for Triplanner to investigate a solution in a larger search domain as the slower SUAS speeds result in a tighter turn radius allowing the vehicle to traverse through highly constrained regions. The rate limited acceleration vector is then formed based on the change in the velocity curve. These vectors are shown in Figure 5.

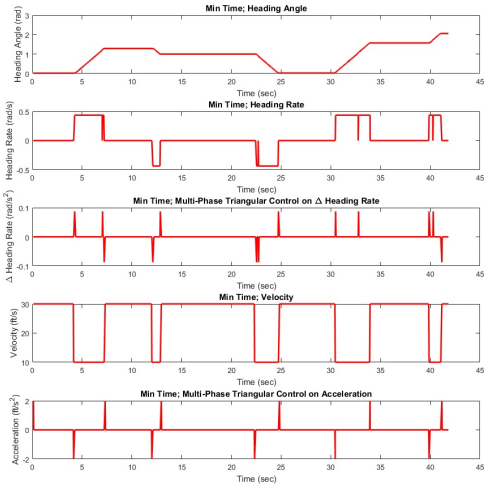

Figure 5. Initial State and Control Vectors

Each of these vectors are described in a three tier structure representing the defined interval and phase of the solution. This process is repeated for each interval with the structure being appended to the end of the previous interval. Once all target intervals have been exhausted, the connectivity and guess structures are output to the optimal control software.

Optimal Control Problem

The optimal control problem is initialized by first establishing the start and end locations of the current interval and building three structures for input to the NLP solver. The first structure defines the bounds on the states, control, and time. The state vector represents the position in barycentric coordinates, heading angle, heading angle rate, and the SUAS velocity defined by the dynamic equations

$$\dot{\alpha}_1^{(p)}(t) = \frac{(y_2-y_3)(v(t))cos\theta(t)+(x_3-x_2)(v(t))sin\theta(t)}{det(T)} \quad (15)$$

$$\dot{\alpha}_2^{(p)}(t) = \frac{(y_3-y_1)(v(t))cos\theta(t)+(x_1-x_3)(v(t))sin\theta(t)}{det(T)} \quad (16)$$

$$\dot{\alpha}_3^{(p)}(t) = -\dot{\alpha}_1^{(p)}(t) - \dot{\alpha}_2^{(p)}(t) \quad (17)$$

$$\dot{\theta}^{(p)}(t) = \theta^{(p)}(t) \quad (18)$$

$$\ddot{\theta}^{(p)}(t) = \gamma^{(p)}(t) \quad (19)$$

$$\dot{v}^{(p)}(t) = a^{(p)}(t) \quad (20)$$

$\forall p \in [1...P]$, resulting in the state vector

$$\mathbf{X} = (\alpha_\mathbf{1}, \alpha_\mathbf{2}, \alpha_\mathbf{3}, \theta, \dot{\theta}, \mathbf{v}). \quad (21)$$

The control for the SUAS is on the change in heading rate and acceleration resulting in the control vector

$$\mathbf{u} = (\gamma, \mathbf{a}). \quad (22)$$

Bounds are applied on the states, control, and time to limit the search domain of the NLP solver. The bounds on the first three position states are enforced with the start and end location of each interval

$$[\alpha_1, \alpha_2, \alpha_3]^{(1)} = start\_position \quad (23)$$

$$[\alpha_1, \alpha_2, \alpha_3]^{(P)} = end\_position \quad (24)$$

with the intermediate phases for the first three states bounded by

$$0 \leq \alpha_1^{(p)} \leq 1 \quad (25)$$

$$0 \leq \alpha_2^{(p)} \leq 1 \quad (26)$$

$$0 \leq \alpha_3^{(p)} \leq 1 \quad (27)$$

$\forall p \in [2...P-1]$ within each interval. The bounds for the remaining states, control, and time within each interval are enforced as

$$|\theta^{(p)}| \leq \pi \quad (28)$$

$$|\dot{\theta}^{(p)}| \leq 25 \ \ deg/s \quad (29)$$

$$|\gamma^{(p)}| \leq 5 \ \ deg/s^2 \quad (30)$$

$$10 \ \ ft/s \leq v^{(p)} \leq 30 \ \ ft/s \quad (31)$$

$$|a^{(p)}| \leq 2 \ \ deg/s^2 \quad (32)$$

$$0 \leq t^{(p)} \leq \frac{edge_{max}^{(p)}}{v} \quad (33)$$

$\forall p \in [1...P]$ and where $edge_{max}$ represents the longest edge in the defined simplex.

The second structure defined for each interval enforces the first mesh along with the number of collection points used per mesh interval within each phase. For this work, these values remain consistent across each waypoint interval and each phase and are defined previously in Table 1.

The third structure builds the event constraints required for the continuous transition of the state and control across each phase boundary within the current interval defined by

$$X_o^{(p)} - X_f^{(p-1)} = 0 \ \ \forall p \in [2...P]. \quad (34)$$

Since the initial guess is evaluated individually for each interval, a discontinuity could exist in the Triplanner solution at each waypoint. To eliminate this discontinuity in the optimal solution, an event constraint between each interval must be established to assure a continuous transition of the states. The requirement to maintain the position location between intervals is enforced in Equations 23-24, with the remaining states enforced in the event constraint here

$$X_o(4:6)^{(\delta)} - X_f(4:6)^{(\delta-1)} = 0 \;\; \forall \delta \in [2...\Delta] \quad (35)$$

where δ represents the current interval and Δ defines the total number of intervals in the solution.

This process is repeated until all waypoint intervals have been exhausted. The interval structure for the bounds, mesh, initial guess, and events are then combined into a single structure with the number of phases, P_t, equal to the sum of the phases in each waypoint interval defined by

$$p_t = [p^{\delta_1}; p^{\delta_2}; \ldots; p^{\delta_\Delta}] \quad (36)$$

$\forall p \in [1...P]$ and $p_t \in [1...P_t]$.

Finally, the objective of this optimal control problem is to solve the minimum flight path over all phases subject to the dynamic constraints defined in Equations 15-20, parameter bounds of Equations 23-33, and event constraints of Equations 34-35. The cost function is defined as

$$J^{(p_t)} = \int_{t_0^{(p_t)}}^{t_f^{(p_t)}} dt \;\; \forall p_t \in [1...P_t] \quad (37)$$

$$J = \sum_{p_t=1}^{P_t} J^{(p_t)}. \quad (38)$$

5. Scenario

As in [7] and [8], the scenario presented considers the two-dimensional flight of a SUAS through an urban environment modeled in downtown Chicago, USA. Buildings that reach an altitude greater than 550 feet Above Ground Level (AGL) are modeled as polygonal constraints. Given a starting location, the aircraft is required to fly over three separate waypoints before returning to the start location. This results in a four interval solution. For the work herein, the required start and waypoint locations are defined as

$$start = [200, 200]ft \quad (39)$$
$$waypoint_1 = [1825, 1700]ft \quad (40)$$
$$waypoint_2 = [2370, 2355]ft \quad (41)$$
$$waypoint_3 = [3650, 1215]ft. \quad (42)$$

Figure 6 shows the building constraints as red polygons, and the start and waypoint locations as the black asterisks.

The aircraft is required to maintain an altitude of 600 feet AGL while avoiding all building constraints. The Triplanner algorithm is initiated with the start and end location for each interval as well as the minimum turn radius, R, defined by the vehicles minimum velocity and turn rate limit.

$$R = \frac{v_{min}}{\dot{\theta}_{max}} \quad (43)$$

The SUAS maintains control on the change in heading rate and acceleration, thus allowing varying rate turns and the

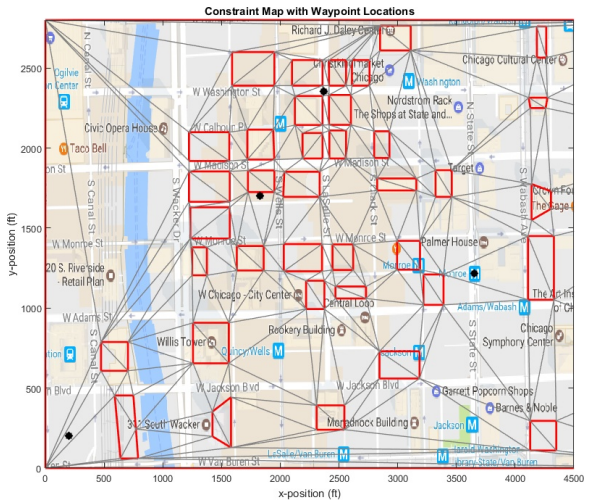

Figure 6. Constraint Map. Map Data @2017 Google

potential to maximize the search domain. The waypoints were chosen to illustrate the vehicle control and provide for a challenging optimal control problem. The first waypoint is located on the front doorstep of the associated building, requiring the SUAS to fly adjacent to the constrained edge followed by a sharp turn to the north of the city. The second waypoint monitors the center of an intersection surrounded by four constraints on each corner. Finally, the third waypoint is located at a metro station designed to extend the four interval problem through a large number of simplexes.

6. Results

The Triplanner algorithm is used to determine four individual path solutions, representing each defined interval. Figure 7 shows the connected four interval solution and the defined polygonal search channel.

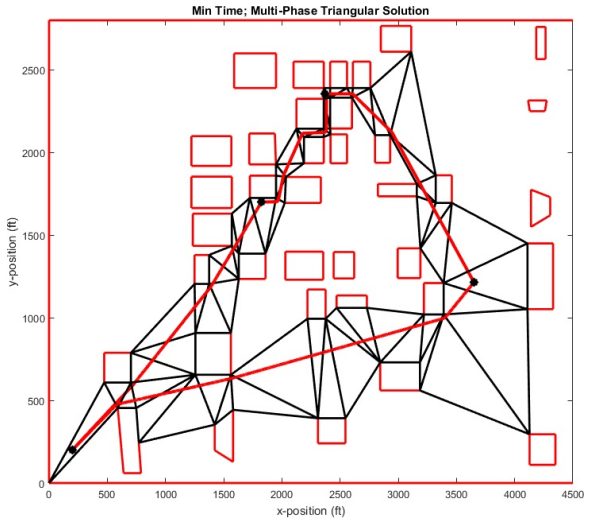

Figure 7. Triplanner Solution. Map Data @2017 Google

This geometric path solution is shown with the solid red line and avoids the polygonal path constraints with a radial distance calculated from the minimum flight speed as defined in Equation 43. The path contained within each interval

results in a sub-optimal Dubins path solution as the minimum radius turns are not optimally located due to the constrained field and the formulation of the Triplanner algorithm. Additionally, the heading rate limits are not upheld at the waypoint locations resulting in a discontinuity in the vehicle heading angle. The search channel for each interval is shown with the black outlined polygons and are completely maintained outside of all path constraints. This provides a search domain for the optimal control solver free of all building constraints. Table 2 shows the number of phases, flight time, and computation time for each of the intervals of the Triplanner solution.

Table 2. Triplanner Interval Solutions

Interval	Phases	Flight Time (s)	Computation Time (ms)
1	17	74.16	7.37
2	15	41.84	5.96
3	18	61	4.78
4	23	123.6	6.89

The combined path results in 73 phases with a total flight time of 300.6 seconds.

The initial guess for the optimal control solver can now be formed by combining the Triplanner solutions of the four intervals into appropriately formatted structures. Figure 8 shows the optimal path contained within the defined search channel.

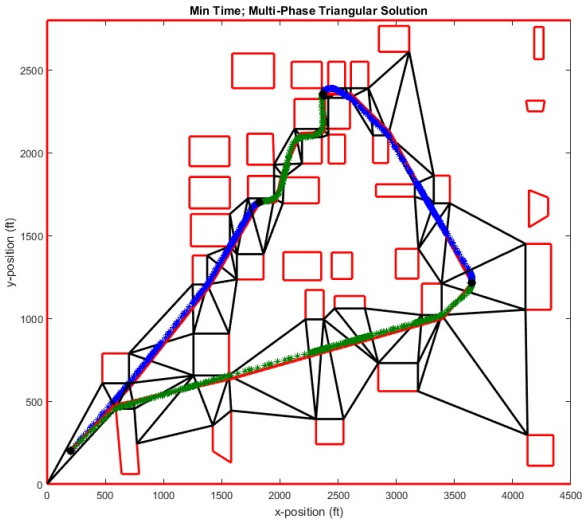

Figure 8. **GPOPS-II Optimal Solution. Map Data @2017 Google**

The optimal path is color coded with blue and green asterisks to illustrate the different intervals of the problem. The transition between intervals is continuous and rate limited according to the limitation of the problem parameters and enforced through the event constraints of the optimal control problem described in Equations 34 and 35. The path solution is optimized over the Dubins path result from Triplanner as the turn points are moved to an optimal location around the building constraints allowing for a more direct path to next waypoint. The final path solution is comprised of 73 phases with an objective time of 290.51 seconds, a 10 second improvement from the initial guess. The required computation time for the optimal solution was 117.45 seconds on a PC.

The state vectors for the heading, heading rate, and velocity as well as the control vectors are shown in Figure 9.

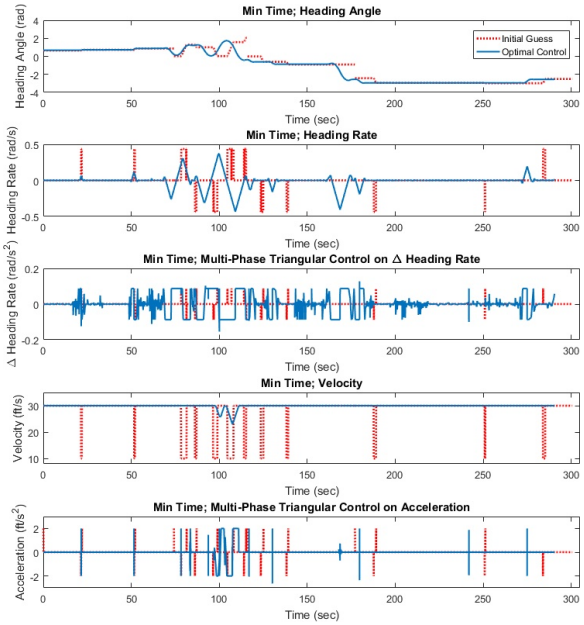

Figure 9. **Optimal State and Control Vectors**

The initial guess, formed from the result of the Triplanner solution, is shown with the dashed red line while the optimal state and control vectors are shown with the solid blue lines. Evaluating the heading angle in the first chart, the discontinuities in the Triplanner heading angle can be seen at each waypoint, located at 74, 116, and 177 seconds respectively. The optimal heading rate resembles the turn points of the Triplanner solution, but is solved slightly faster than the initial guess provided. The second chart of Figure 9 shows the heading rate. The initial guess provided maximum rate turns based on the Dubins path solution of Triplanner. The optimal heading rate was limited on the vehicles turns due to the rate-limited control vector shown in the third chart. Finally, the velocity and acceleration vectors are shown in charts four and five respectively. The control on the vehicles acceleration is illustrated in the velocity vector as the vehicle is required to slow down in order to complete turning requirements designed at the second waypoint.

7. CONCLUSIONS

The fast computation times of these geometric solutions provide a foundation for achieving real-time, onboard operations with optimal control software. The Triplanner algorithm successfully provided a triangular mesh, a search channel free of path constraints, and an initial solution for the SUAS position states. However, since Triplanner provides a point to point solution, the intervals were solved individually creating a discontinuity in the heading angle of the estimated solution. This creates the requirement for optimal control software as the 2010 version of the Triplanner algorithm does not account for initial or final heading angle constraints or rate limited control inputs. The initial path solution was evaluated and heading angle rates, velocity, and acceleration vectors were established. These vectors were implemented in the optimal control solver, GPOPS-II for calculation of the optimal path within the desired polygonal search channel.

By defining the problem in barycentric coordinates, implementation of the Triplanner solution as the initial guess was accomplished through a phased approach where each simplex represented a single phase of the optimal solution. A continuous transition of the states and control were accomplished through event constraints, allowing the optimal solution to continuously transition from one interval to the next across each waypoint. Further, the challenging urban environment presented tight corridors between building constraints where the vehicle was required to slow its airspeed in order to reduce the minimum turning radius required to achieve the desired path. By providing control to the vehicle's acceleration, minimal speed deviations were realized and a continuous minimum time solution was achieved over four intervals containing three waypoints. The computation times for the optimal control problem exceeded the limits for real-time operations, but the simulation presented provides a foundation for future work where computational times could be drastically reduced by determining the appropriate limits for a series of finite-horizon optimal control problems combined to meet the same objective posed herein.

ACKNOWLEDGMENTS

The authors would like to thank the Air Force Research Laboratory, Aerospace Systems Directorate, Power and Controls Division, who sponsored the work and provided the challenging problem concept, background information, and continual support.

REFERENCES

[1] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture," *IEEE Transaction on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[2] M. Ataei and A. Yousefi-koma, "Three-dimensional optimal path planning for waypoint guidance of an autonomous underwater vehicle," *Robotics and Autonomous Systems*, vol. 67, pp. 23–32, 2015.

[3] "Air Force Research Laboratory Autonomy Science and Technology Strategy," Tech. Rep., 2013.

[4] N. M. Jodeh and R. G. Cobb, "Optimal Airborne Trajectories for Data Collection from Wireless Sensor Networks by Direct Collocation Methods ," in *AIAA Guidance, Navigation, and Control Conference*, Kissimmee, Fl, 2015.

[5] A. Kaplan, N. Kingry, P. Uhing, and R. Dai, "Time-Optimal Path Planning With Power Schedules for a Solar-Powered Ground Robot," *IEEE Transactios on Autonomous Science and Engineering*, vol. 14, no. 2, pp. 1235–1244, 2017.

[6] M. D. Zollars and R. G. Cobb, "Simplex Methods for Optimal Control of Unmanned Aircraft Flight Trajectories," in *Proceedings of the ASME 2017 Dynamics Systems and Controls Conference*, 2017, p. 10.

[7] M. D. Zollars, R. G. Cobb, and D. J. Grymin, "Simplex Solutions for Optimal Control Flight Paths in Urban Environments," *Journal of Aeronautics and Aerospace Engineering*, vol. 6, no. 3, p. 8, 2017.

[8] ——, "Simplex Optimal Control Methods for Urban Environment Path Planning," in *To be published in the proceedings of the AIAA Sci-Tech Conference, 2018*, Orlando, FL, 2018, p. 16.

[9] D. Garg, "Advances In Global Psuedospectral Methods For Optimal Control," Dissertation, University of Florida, 2011.

[10] M. Zollars, P. Blue, and B. Burns, "Wind Corrected Flight Path Planning for Autonomous Micro Air Vehicles Utilizing Optimization Techniques," in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, Hilton Head, SC, 2007.

[11] C. J. Humphreys and R. G. Cobb, "A Hybrid Optimization Technique Applied to the Intermediate-Target Optimal Control Problem," *Global Journal of Technology and Optimization*, vol. 7, no. 2, 2016.

[12] P. T. Heidlauf, "Optimal UAV Path Planning with Dynamic No-Fly-Zones for Target Geolocation using Line-of-Bearing Measurements and Kalman Filtering," MS thesis, Air Force Institute of Technology, 2017.

[13] M. Kapadia and N. I. Badler, "Navigation and steering for autonomous virtual humans," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 4, no. 3, pp. 263–272, 2013.

[14] M. Kallmann, "Shortest Paths with Arbitrary Clearance from Navigation Meshes," in *Proceedings of the Eurographics SIGGRAPH Symposium on Computer Animation SCA*, 2010.

[15] ——, "Dynamic and Robust Local Clearance Triangulations," *Acm Transactions on Graphics*, vol. 33, no. 5, p. 17, 2014.

[16] ——, "Navigation Queries from Triangular Meshes," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6459 LNCS, pp. 230–241, 2010.

[17] M. Meyer, A. Barr, H. Lee, and M. Desbrun, "Generalized Barycentric Coordinates on Irregular Polygons," *Journal of Graphics Tools*, vol. 7, no. 1, pp. 13–22, 2002.

[18] M. A. Patterson and A. V. Rao, "GPOPS-II manul: A General-Purpose MATLAB Software for Solving Multiple-Phase Optimal Control Problems Version 2 . 1," no. October, pp. 1–72, 2015.

BIOGRAPHY

Michael D. Zollars *received his BS in Mechanical Engineering from the Pennsylvania State University in 2003 and his MS in Aeronautical Engineering from the Air Force Institute of Technology in 2007. He is currently a PhD candidate at AFIT with research focused in optimization of aircraft path planning for autonomous systems.*

Richard G. Cobb *received a B.S. in Aerospace Engineering from the Pennsylvania State University in 1988 and an M.S. and PhD in Astronautical Engineering from the Air Force Institute of Technology in 1992 and 1996 respectively. He is a Professor with the Department of Aeronautics and Astronautics at the Air Force Institute of Technology. His research interests include Optimal Control Theory and Applications and Spacecraft Systems.*

David J. Grymin *received his BS and MS in Mechanical Engineering from Rochester Institute of Technology in 2009. He received the PhD degree in Aerospace Engineering in 2013 from Virginia Tech. He is currently a research aerospace engineer with the Air Force Research Laboratory, Wright-Patterson Air Force Base, OH.*

Appendix F. American Controls Conference 2018

Appendix F contains the fifth paper published in this research effort. This work augments the cost function to minimize incursions to keep-out regions within the constrained map of downtown Chicago, USA. It was published and presented at the AIAA American Controls Conference, Milwaukee, Wisconsin, in June 2018.

Optimal Path Planning for SUAS Target Observation through Constrained Urban Environments using Simplex Methods

Michael D. Zollars[1], Richard G. Cobb[2], and David J. Grymin[3]

***Abstract*— The work herein determines the optimal flight path for a Small Unmanned Aircraft System through a constrained urban environment while minimizing the flight through keep-out regions and ending on a defined orbit around a target of interest. Direct orthogonal collocation methods are combined with fast geometric path planning techniques where a triangulated mesh is used to produce a hybrid control routine resulting in optimal flight paths through a defined triangulated channel. Physical constraints are eliminated from the non-linear program search space, while keep-out regions are modeled within the objective function of the optimal control problem and avoided according to a weighted distribution of the objective components. A scenario is presented for a SUAS to advance at constant altitude through city building constraints while minimizing time in unavoidable keep-out regions. The path terminates outside the triangulated channel on an orbit, encircling the target location. Results illustrate path constraints designed in the objective functional within the construct of a triangulated mesh and the implications that result.**

I. INTRODUCTION

Small Unmanned Aircraft Systems (SUAS) have played a major role in mission contributions across the Department of Defense (DoD). Their impact continues to be recognized as their role transitions from remote piloted operations and waypoint following to autonomous navigation. Manned Unmanned Teaming (MUM-T) has become a research priority with goals set to demonstrate effective humane-machine interactions while increasing the trust of autonomous missions [1]. These SUAS, performing autonomously, have the potential to utilize sensor packages in close proximity to areas of interest and provide reliable data to manned aircraft located at a safe off-set distance.

Within the construct of autonomous flight path planning, direct orthogonal collocation methods have significantly increased computation efficiencies when an adequate initial guess is used to seed the Nonlinear Programming (NLP) solver and parameter bounds are accurately represented [2], [3], [4], [5]. Leveraging computer animation algorithms, the optimal control problem can be formulated within a triangulated mesh utilizing a barycentric coordinate system, thus standardizing parameter bounds within each simplex. Further, the triangulated mesh eliminates nonlinear path constraints from the search space and generate a Dubins path solution through a triangulated channel, producing an adequate solution for seeding the NLP. Previous work has shown the difficulties in efficiently modeling a large number of constraints within the NLP while illustrating the effectiveness of providing the solver an initial guess posed through a triangulated mesh [6]. Additional work has illustrated the effectiveness and efficiencies gained by removing constraints from the search domain of the optimal control control problem by applying a simplex structure [7], [8].

The scenario presented in this work addresses the case where a path constraint cannot be completely avoided and therefore the objective function must account for minimizing the time spent in the keep-out region. Further, the vehicle is required to end on a path coincident to a final desired orbit with a heading vector perpendicular to the target location. The terminal phase of the scenario requires transition from barycentric coordinates to the feasible regions in global Cartesian coordinates to allow for an optimal trajectory to the final orbit which may pass through multiple simplexes.

II. BACKGROUND

A. Direct Orthogonal Collocation

Many methods have been developed to transcribe an infinite-dimensional optimal control problem into a finite optimal control problem, or NLP. A few of these include shooting methods [9], state and control parameterization methods [10], and direct orthogonal collocation methods [2], [11]. Direct orthogonal collocation methods are accomplished by first approximating the continuous functions of the optimal control problem with a finite dimensional Lagrange polynomial basis where the state vector is approximated at a set of collocated points described as

$$\tilde{x}(\tau) \approx \tilde{x}_N(\tau) = \sum_{i=1}^{n+1} x_i L_i(\tau). \tag{1}$$

Here, x_i represents the weight function, $L_i(\tau)$ is the Lagrange polynomial basis described as

$$L_i(\tau) = \prod_{\substack{i=0 \\ j \neq i}}^{n} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{2}$$

and τ represents an affine transformation of the time t on the interval from $(-1, 1)$. The desire is to determine the solution at each collocated point while including the initial point in the mesh, defined as Legendre-Gauss-Radau points. Differentiation or integration of the state and control is then calculated through Gaussian quadrature. This

[1]Michael D. Zollars is a PhD Candidate in the Department of Aeronautics and Astronautics, Air Force Institute of Technology, Wright-Patterson AFB, OH, 45433

[2]Richard G. Cobb is a Professor of Aeronautics and Astronautics, Air Force Institute of Technology, Wright-Patterson AFB, OH, 45433

[3]David J. Grymin is a Controls Science Engineer at the Control Science Center of Excellence, Air Vehicles Directorate, Air Force Research Laboratory, Wright-Patterson Air Force Base, OH 45433

method is termed *global* as each collocation point is solved simultaneously. However, in order to fully take advantage of the efficiencies of Lagrange polynomials and Gaussian quadrature, a quality guess of the states, control and time must be provided to allow for efficient convergence of the optimal control problem.

B. Constraint Models

The complexity of the optimal control problem can grow exponentially when constraint models are incorporated. These models can range from simplistic shapes representing circular or elliptical regions, to superquadrics, or even polygonal shapes [6]. Each of these must be modeled as a path constraint in the optimal control problem, reducing the sparsity of the Jacobian matrix and increasing the computational requirements. Further, these path constraints must be smooth differentiable functions in order to quickly acquire the Jacobian and Hessian. This can be problematic when designing algorithms to handle multiple constraints in a timely fashion.

For problems where the constraints must be included in the optimal control problem, previous work [2], [12] has shown the benefits of combining superellipse shapes with sigmoid functions. The superellipse is defined by

$$F(x,y) = \left(\frac{x}{a}\right)^N + \left(\frac{y}{b}\right)^M \tag{3}$$

where any point on the SUAS trajectory is outside of the constraint when $F > 1$. Variables a and b represent the semi-major and semi-minor axes of the superellipse and $N \geq 2$, $M \geq 2$ are even numbers representing the curvature of the shape. These shapes are beneficial in the flexibility to model general constraint structures, however, as N and M are increased, the gradient of the function becomes excessively large at the corners and the function itself can grow without bound. To minimize the impacts this can have on an NLP solver, the function is incorporated into a modified inside-outside function through a sigmoid function [12]. This allows for a bounded, continuous, and differentiable function.

$$\phi(F) = \frac{1}{1 + e^{(s(F(x,y)-1))}} \tag{4}$$

Here, s represents the stiffness parameter of the curve. Figure 1 shows a straight line vehicle trajectory, annotated with the blue asterisks, through a constrained circular region. The gray curves depict the stiffness of the function as s is varied from 0.1 to 10. Incorporating this function as a path constraint, any functional value greater than zero represents a position close to or inside the keep-out region dependent on the stiffness parameter. The normalization of this function value can be handled through the distributed weight values when included in a cost function. This work implements a stiffness value of $s = 2$, illustrated by the solid black line in Figure 1.

C. Virtual Environments

When considering the constrained, constant altitude SUAS flight trajectory problem, parallels can be made to path planning techniques used in computer animation. Virtual worlds

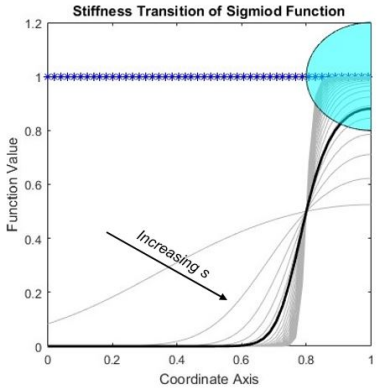

Fig. 1. Stiffness Transition of Sigmoid Function

are populated with autonomous agents that are required to traverse through simulated environments while avoiding all obstacles. The final result must be computationally efficient while taking into account path length, time, and energy expended to produce a realistic simulation operated in a real-time environment [13].

An extensive review of path planning though virtual environments with clearances is given in [14]. Algorithms to determine shortest paths while providing a minimum clearance to all constraints are provided in [15]. These algorithms focus on providing reliable, computationally efficient path solutions and provide a framework for dynamically removing or adding constraint regions. The work herein implements these algorithms in the 2010 version of the Triplanner toolkit[1]. A more extensive review of the algorithm can be found in [14], [16].

III. Methodology

A. Initial Path Solution

The Triplanner algorithm is initiated with the start and final coordinates, the polygonal constraints, and the required clearance distance from each constraint. To equate the autonomous agent to a SUAS in the two-dimensional plane, the clearance distance is set to the minimum turning radius of the aircraft, computed based on a bank angle limitation at constant speed. The output of the Triplanner algorithm consists of all the constrained and unconstrained edges, a defined simplex channel free of constraints, and a Dubins path solution contained within the simplex channel in global Cartesian coordinates.

The Triplanner algorithm can navigate through over $60K$ constraints on the order of milliseconds [14], however, limitations to the algorithm do exist. First, Triplanner is strictly a path planning algorithm without regard for limitations on system parameters and control. Second, keep-out regions that are unavoidable cannot be handled with the Triplanner algorithm as the triangulated mesh eliminates all constraints from the search space. Finally, Triplanner provides a point to point solution and does not satisfy the terminal requirement

[1] http://graphics.ucmerced.edu/software/tripath/

to optimally end the path on a non-deterministic point of the desired orbit around the target. These issues must be overcome before the initial guess is provided to the NLP.

The initial guess for the state solution, time, and control through a triangulated mesh is extensively covered in previous work [7]. To assure the desired orbit radius around the target remains in feasible space, a line intersection algorithm is used to first determine the distance from the target location to the closest constrained edge, providing an upper bound to the orbit radius. This is determined by finding the perpendicular distance from the target point to each constrained line segment. Constructing Bourke's algorithm [17], the slope, λ, of the perpendicular line segment is defined as

$$\lambda = \frac{(x_t - x_1)(x_2 - x_1) + (y_t - y_1)(y_2 - y_1)}{\|(x_2, y_2) - (x_1, y_1)\|_2^2} \tag{5}$$

where (x_t, y_t) defines the target location and (x_1, y_1), (x_2, y_2) define the vertex points of the constrained edge. A line equation is used to determine the point of intersection between the perpendicular line and the constrained line vector,

$$x_3 = x_1 + \lambda(x_2 - x_1) \tag{6}$$

$$y_3 = y_1 + \lambda(y_2 - y_1). \tag{7}$$

The shortest distance from the target point to the constrained edge, ζ_k is then defined by

$$\zeta_k = min(\|(x_t, y_t) - (x_i, y_i)\|_2) \forall i \in [1, 2, 3]. \tag{8}$$

Evaluating Equation 8 for each constrained edge, the maximum orbit radius for the SUAS is defined by the minimum distance to the closest constrained edge in the domain,

$$R_{max} = min(\zeta_k) \forall k \in [1...C] \tag{9}$$

where C defines the total number of constrained edges. This process is shown in Figure 2 for a single constrained edge.

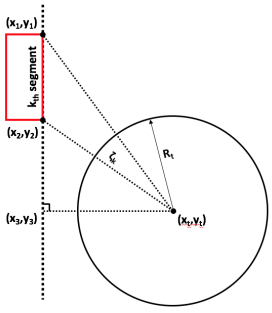

Fig. 2. Maximum Orbit Radius

Finally, the R_{max} value is checked against the required orbit radius, R_t, to assure the SUAS remains in an unconstrained, feasible airspace while accomplishing its mission. The final orbit radius is then used to determine the intersection of simplexes, which are removed from the search channel. This allows for the NLP solver to converge to an optimal solution without the requirement of traversing through a specific simplex.

The initial guess from the final simplex phase through the orbit consists of a straight line solution to the intersection of the orbit radius. This line is determined by defining the unit vector from the final position of the last simplex of the Triplanner solution to the target point. A line intersection algorithm is then used to find the transition point from the straight line path to a circular orbit based on the length of the target radius and the vehicles minimum turning radius.

B. Optimal Control Problem

Within the construct of a triangulated mesh, the optimal control problem is formulated into a phased approach. Each simplex that is traversed is represented as a single phase p, and each phase is connected through event constraints equating the states, control and time at each boundary. The total number of phases is represented as P. The simplex corridor is defined through the results of the Triplanner algorithm and account for phases $1 : N - 1$. The final phase is evaluated in global coordinates to allow for left or right turn orbits of varying radius around the target. The SUAS is defined with 2-dimensional dynamics in a three-state model defined as

$$\dot{x}(t) = (v)cos(\theta(t)) \tag{10}$$

$$\dot{y}(t) = (v)sin(\theta(t)) \tag{11}$$

$$\dot{\theta}(t) = \gamma(t). \tag{12}$$

Here, the velocity, v, is held constant at $30 ft/sec$ through the entirety of this work. The state vector is defined as

$$\tilde{X} = (x, y, \theta) \tag{13}$$

with control

$$u = \gamma. \tag{14}$$

Transforming the coordinates into a barycentric coordinate frame and taking the derivative of the weight equations with respect to the x and y coordinates, a new set of dynamic equations can be defined to propagate the state variables through each simplex. This results in an equivalent set of dynamics defined through the barycentric weights, previously described in [6] as

$$\dot{\alpha}_1^{(p)}(t) = \frac{(y_2 - y_3)(v)(cos\theta(t)) + (x_3 - x_2)(v)(sin\theta(t))}{det(T)} \tag{15}$$

$$\dot{\alpha}_2^{(p)}(t) = \frac{(y_3 - y_1)(v)(cos\theta(t)) + (x_1 - x_3)(v)(sin\theta(t))}{det(T)} \tag{16}$$

$$\dot{\alpha}_3^{(p)}(t) = -\dot{\alpha}_1^{(p)}(t) - \dot{\alpha}_2^{(p)}(t) \tag{17}$$

$$\dot{\theta}^{(p)}(t) = \gamma(t) \tag{18}$$

$\forall p \in [1...P - 1]$, with the control consistent with (14), and the new state vector defined as

$$X = (\alpha_1, \alpha_2, \alpha_3, \theta). \tag{19}$$

With the dynamics described in terms of a weighted measure to each simplex vertex, a phased solution can be constructed where each simplex represents a single phase in the optimal control problem.

By transforming the problem to the barycentric coordinate frame, parameter bounds on the state can be generalized

for each phase by taking advantage of the coordinate frame properties defined as

$$0 \leq \alpha_1^{(p)} \leq 1 \tag{20}$$
$$0 \leq \alpha_2^{(p)} \leq 1 \tag{21}$$
$$0 \leq \alpha_3^{(p)} \leq 1. \tag{22}$$

Parameter bounds on the heading, control, and time are given as

$$|\theta^{(p)}| \leq 180 \ \ deg \tag{23}$$
$$|u^{(p)}| \leq 25 \ \ deg/s \tag{24}$$
$$0 \leq t^{(p)} \leq \frac{(\rho) edge_{max}^{(p)}}{v}, \tag{25}$$

where $edge_{max}$ represents the longest edge in the defined simplex and $\rho > 1$ defines a weighting to increase the time bound to allow for curved solutions through a simplex, accounting for potential flight through keep-out regions. Bounds for the final phase are determined by the maximum and minimum values of the final simplex edge and the orbit around the target.

Finally, event constraints are imposed to maintain a continuous transition of the state variables through each simplex,

$$X_o^{(p)} - X_f^{(p-1)} = 0 \ \ \forall p \in [2...P]. \tag{26}$$

The final phase requires the vehicle to end on a constant radius orbit around the target. This phase takes place outside of the triangulated search channel requiring the implemented dynamics to be in global Cartesian coordinates represented in Equations 10, 11, and 12 with control as the heading rate, γ. The event constraint relating phase $P-1$ to P must be equated through a transformation of the coordinate states of the $P-1$ phase as follows,

$$x_f^{(P-1)} = \alpha_{1f}^{(P-1)} x_1 + \alpha_{2f}^{(P-1)} x_2 + \alpha_{3f}^{(P-1)} x3 \tag{27}$$
$$y_f^{(P-1)} = \alpha_{1f}^{(P-1)} y_1 + \alpha_{2f}^{(P-1)} y_2 + \alpha_{3f}^{(P-1)} y3, \tag{28}$$

where x_i and y_i represent the simplex vertex locations. This results in the event constraint

$$\tilde{X}_o^{(P)} - \tilde{X}_f^{(P-1)} = 0. \tag{29}$$

This constraint bridges the state and control of the final simplex edge to the final phase. To account for the terminal condition, a final event constraint is added to implement the tangency condition to the final orbit. The tangency condition is defined for the final x and y positions as

$$\tilde{x}_f = x_t + R_t cos(\theta_f + \mu \tfrac{\pi}{2}) \tag{30}$$
$$\tilde{y}_f = y_t + R_t sin(\theta_f + \mu \tfrac{\pi}{2}), \tag{31}$$

where $\tilde{x}_f$ and $\tilde{y}_f$ represent a vehicle position coincident and perpendicular to the desired final orbit of center x_t, y_t with radius R_t while $\mu \in [-1, 1]$ implements a clockwise or counter-clockwise orbit dependent on the vehicles sensor location. The final event constraint is applied to the first two states described as

$$\tilde{x}_f - x_f^P = 0 \tag{32}$$
$$\tilde{y}_f - y_f^P = 0. \tag{33}$$

With the dynamics and parameter bounds defined, the objective function is designed for minimum time of flight, while including penalties for the keep-out zones. The cost associated with the minimum time flight through each phase is represented as

$$J_{minT}^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} dt \ \ \forall p \in [1...P]. \tag{34}$$

The penalty for the keep keep-out regions, $F_i(x, y)$ is defined with a sigmoid function, $\phi(F)$, and minimizes the keep-out incursions at each collocation point as follows

$$F_i(x,y) = \frac{(x^{(p)}(t) - Kx_i)^2}{a_i} + \frac{(y^{(p)}(t) - Ky_i)^2}{b_i} \tag{35}$$
$$\phi_i(F_i) = \frac{1}{1 + e^{(s_i(F_i(x,y) - 1)}}, \tag{36}$$

where Kx_i and Ky_i define the keep-out center point and a_i and b_i define the semi-major and semi-minor axis. This yields a cost function, minimizing the incursion onto keep-out regions, defined as

$$J_{minE}^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} \phi_i^{(p)}(F_i) dt \ \ \forall p \in [1...P-1] \tag{37}$$

The complete objective function is a summation of the minimum time cost and minimum incursion to keep-out regions.

$$J = \sum_{p=1}^{P} \beta_1 J_{minT}^{(p)} + \sum_{p=1}^{P-1} \beta_2 J_{minE}^{(p)} \tag{38}$$

Here, β_i defines the weight values that sum to unity and distribute the cost proportionally, such that the components of the cost function influence the desired flight path.

With the cost function defined, integral bounds are implemented within the optimal control problem as follows,

$$0 \leq J_{minE} \leq 50. \tag{39}$$

The optimal control problem is solved using the General Purpose Optimization Psuedospectral Software (GPOPS-II). The parameters used in the simulation are shown in Table I.

TABLE I
GPOPS-II USER SETTINGS

GPOPS-II User Settings	
Mesh Method	hp-PattersonRao
Mesh Tolerance	10^{-2}
NLP Solver	SNOPT
Derivative Supplier	SparseCD
Method	RPM-differential
NLP Tolerance	10^{-5}
Min Collocation Points	4
Max Collocation Points	10
Mesh Fraction	$\frac{1}{2}$*ones(1,2)
Mesh collocation Points	4*(1,4)

IV. SCENARIO

Consider a city map representing an urban environment, where an aircraft is required to maintain constant altitude and fly from an initial position to a final region, culminating on an orbit around a target of interest. This scenario is represented

with a constraint map of downtown Chicago, USA. The SUAS is required to maintain an altitude of 600 ft Above Ground Level (AGL). Each building that exceeds 550 ft AGL is modeled as a constraint that must be avoided. Keep-out regions representing minimal flight zones are included along the West river and down State Street. These zones must be avoided when possible. Finally, the SUAS is required to end the flight path on a circular orbit of the target. Figure 3 illustrates the constraint map for the intended scenario as well as the triangulated mesh over the domain. The initial starting

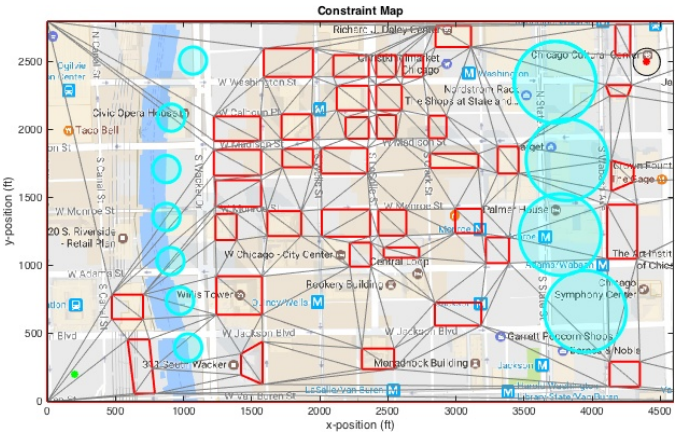

Fig. 3. Downtown Chicago Constraint Map. Map Data @2017 Google

position and the target location are shown with the green and red asterisk respectively. The depicted red polygons outline building constraints that must be avoided while the blue keep-out regions are modeled as circular regions upon which the SUAS must minimize incursions. The left column of keep-out regions have a separation between them, allowing the aircraft to fully avoid the minimal flight zones if they reside completely inside the triangulated search channel. The right column of keep-out regions maintain a small overlap, requiring flight through the zones at a minimum incursion level, dependent on the triangulated search channel. The circle around the target point, located at $(4440ft, 2640ft)$ represents the desired orbit for the SUAS at $R_t = 100$ ft.

V. Results

The initial guess used to seed the NLP is determined using the Triplanner algorithm. The output of Triplanner, shown in Figure 4, consists of a Dubins path solution contained inside a triangulated search channel shown as a series of black simplexes. All building constraints are contained outside of the search channel thus eliminating path constraints from the problem formulation. The initial path guess found by Triplanner is illustrated with the red dashed line and does not account for keep out regions but maintains a clearance from each building equal to the minimum turning radius of the SUAS. Only the simplexes in this search channel are presented to the NLP solver so that a computationally efficient search can be performed for the optimal solution within the defined channel.

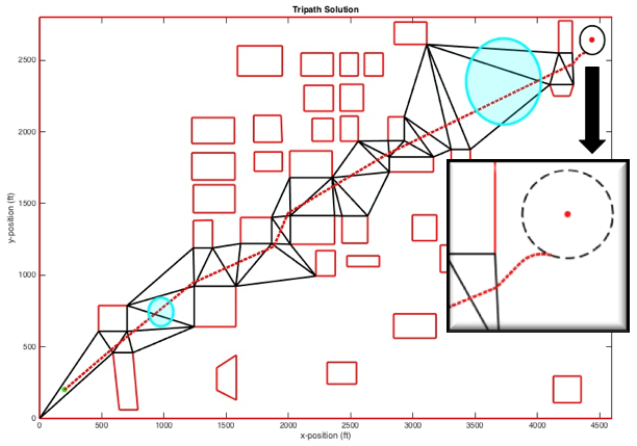

Fig. 4. Tripath Solution ansd Search Corridor. Map Data @2017 Google

Figure 5 shows the resultant optimal control solution given the initial guess from Figure 4 and the user settings defined in Table I. The collocated points of the optimal path through

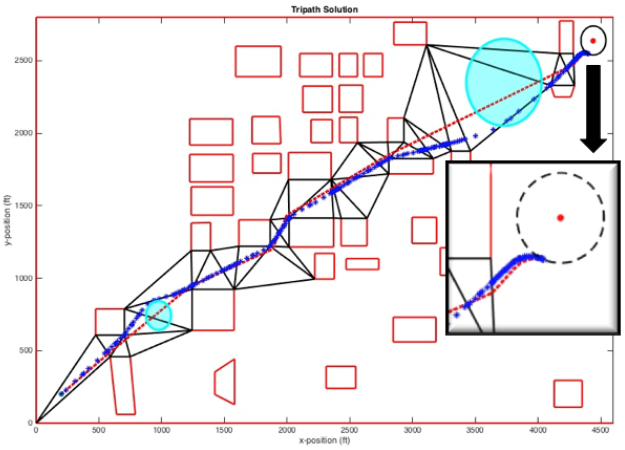

Fig. 5. Optimal GPOPS-II Solution. Map Data @2017 Google

the triangulated search channel are shown with the blue asterisks. The first keep-out region is fully avoided since the region is located completely within the simplex corridor. The second keep-out region extends beyond both boundaries of the simplex channel and therefore the path is dependent on the weighted cost function. The optimal path chosen avoids the keep-out region up to the limit of the triangulated search channel. The final terminal condition is achieved with the SUAS ending on an orbit around the final target.

The vehicle heading and control vectors are shown in the top two plots of Figure 6. The third plot shows the integrated value of the cost function of Equation 37 evaluated over each phase. The deviation in the heading for the first keep-out region can be seen in the heading vector between 20 - 40 seconds into the flight. The lower heading value at the 70 second mark illustrates the benefits gained from the optimal solution over the Dubins path solution. The second keep-out region is illustrated at the 105 second mark of the first plot and the final phase can be seen as a min radius to a

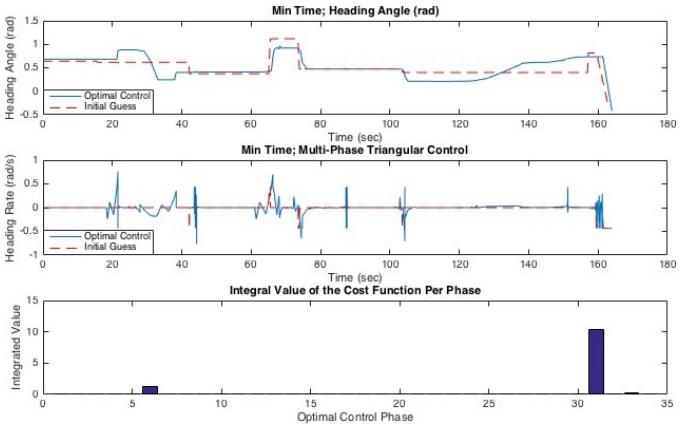

Fig. 6. Optimal Solution Control

tangent location on the final orbit, shown at 160 seconds into the simulation. The integrated values of the third plot show two keep-out regions of which the first only slightly affected the cost function. This value can be adjusted through the stiffness parameter of the sigmoid function described previously. With a value of $s = 2$, a smooth, differential function is evaluated, but impacts to the cost function can be seen when the vehicle approaches a close proximity to the region, while not necessarily entering the region. The second keep-out region is shown to have a greater affect on the cost function, as the region is unavoidable.

VI. Conclusions

The Triplanner algorithm is a computationally efficient algorithm that provides a quality guess and a defined simplex search channel to the NLP solver allowing for the avoidance of hard constraints. In designing the optimal control problem, constraints must be evaluated to determine if they consist of a flight region for which incursions can be minimized, or if it is a constraint that must be avoided completely. If the constraint must be completely avoided, the Triplanner algorithms triangulated mesh will remove the constraint from the search field, limiting the problem's domain. Multiple constraints, combined with the minimum turning radius of the SUAS, could result in Triplanner failing to return a feasible path solution. If however, constraints are modeled as keep-out regions where the SUAS must minimize time spent in a flight zone, the constraint is to be modeled in the optimal control problem rather than the Triplanner algorithm. By modeling the constraint in the cost function, tuning parameters can be used for varying levels of incursions within the keep-out region based on the defined triangulated search channel.

The sigmoid function was shown to be a viable option for modeling constraints in the optimal control cost function within the construct of a triangulated mesh. These functions are appropriate for gradient-based optimization software as they provide smooth, bounded, and differentiable functions. The final cost function was a weighted sum, distributing the cost over flight time and time within keep-out regions. These weights, along with the stiffness parameter of the sigmoid function can be tuned to achieve desired results based on the level of incursion permitted within the keep-out regions.

Ultimately, a discretized simplex mesh was used to provide a foundation for optimal control solutions. When paired with direct orthogonal collocation methods for optimal control, both hard constraints, such as buildings and terrain, as well as keep-out regions, such as unavoidable flight zones, can be modeled and optimal path solutions can be attained.

ACKNOWLEDGMENT

The authors would like to thank the Air Force Research Laboratory, Aerospace Systems Directorate, Power and Controls Division, who sponsored the work and provided the challenging problem concept, background information, and continual support.

References

[1] "Air Force Research Laboratory Autonomy Science and Technology Strategy," Tech. Rep., 2013.

[2] N. E. Smith, "Optimal Collision Avoidance Trajectories for Unmanned/Remotely Piloted Aircraft," Dissertation, Air Force Institute of Technology, 2014.

[3] C. J. Humphreys and R. G. Cobb, "A Hybrid Optimization Technique Applied to the Intermediate-Target Optimal Control Problem," *Global Journal of Technology and Optimization*, vol. 7, no. 2, 2016.

[4] T. J. Masternak, "Multi-Objective Trajectory Optimization of a Hypersonic Reconnaissance Vehicle with Temperature Constraints," Dissertation, Air Force Institute of Technology, 2014.

[5] R. W. Carr and R. Cobb, "An Energy Based Objective for Solving an Optimal Missile Evasion Problem," in *AIAA Guidance, Navigation, and Control Conference*, 2017, pp. 1–18.

[6] M. D. Zollars and R. G. Cobb, "Simplex Methods for Optimal Control of Unmanned Aircraft Flight Trajectories," in *ASME Dynamics Systems and Controls Conference*, 2017, p. 10.

[7] M. D. Zollars, R. G. Cobb, and D. J. Grymin, "Simplex Solutions for Optimal Control Flight Paths in Urban Environments," *Journal of Aeronautics and Aerospace Engineering*, vol. 6, no. 3, p. 8, 2017.

[8] ——, "Simplex Optimal Control Methods for Urban Environment Path Planning," in *AIAA Sci-Tech Information Systems Conference*, Orlando, FL, 2018, p. 16.

[9] J. K. Miller, P. J. Llanos, and G. R. Hintz, "Optimal Control Framework for Impulsive Missile Interception Guidance," in *Guidance, Navigation, and Control Conference*, no. 8, Boston, MA, 2014, pp. 1–14.

[10] J. T. Betts, "Practical methods for optimal control and estimation using nonlinear programming," *Advances in design and control*, p. 434, 2008.

[11] A. W. Suplisson, "Optimal Recovery Trajectories for Automatic Ground Collision Avoidance Systems (Auto GCAS)," Dissertation, Air Force Institute of Technology, 2015.

[12] C. J. Humphreys, R. G. Cobb, D. R. Jacques, and J. A. Reeger, "Dynamic Re-plan of the Loyal Wingman Optimal Control Problem in a Changing Mission Environment," in *AIAA Sci-Tech*, 2016, pp. 1–15.

[13] M. Kapadia and N. I. Badler, "Navigation and steering for autonomous virtual humans," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 4, no. 3, pp. 263–272, 2013.

[14] M. Kallmann, "Shortest Paths with Arbitrary Clearance from Navigation Meshes," in *Proceedings of the Eurographics SIGGRAPH Symposium on Computer Animation SCA*, 2010.

[15] ——, "Dynamic and Robust Local Clearance Triangulations," *ACM Transactions on Graphics*, vol. 33, no. 5, p. 17, 2014.

[16] ——, "Navigation Queries from Triangular Meshes," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6459 LNCS, pp. 230–241, 2010.

[17] P. Bourke, "Points, lines, and planes," 1988. [Online]. Available: http://paulbourke.net/geometry/pointlineplane/

Appendix G. Unmanned Systems Journal 2018

Appendix G contains the sixth paper submitted to the Unmanned Systems journal, July 2018. This work work extends the two-dimensional methodologies to the third dimension and demonstrates the discretization, A* search, and optimal solution for a simple constrained field.

Unmanned Systems, Vol. 0, No. 0 (2013) 1–??

Optimal SUAS Path Planning in Three-Dimensional Constrained Environments

Michael D. Zollars[a,*], Richard G. Cobb[a], David J. Grymin[b]

[a] *Department of Aeronautics and Astronautics, Air Force Institute of Technology, Wright-Patterson AFB, OH, 45433*
E-mail: michael.zollars@us.af.mil and richard.cobb@afit.edu

[b] *Controls Research Engineer, Air Vehicles Directorate, AFRL, WPAFB, OH 45433.*[†]

Small Unmanned Aircraft Systems have grown in autonomy and capability and continue to compliment Department of Defense mission objectives. Teaming unmanned aircraft with manned vehicles can expand mission profiles and reduce risk to human life. To fully leverage unmanned systems, vehicles must be efficient and autonomous in path planning development. The work herein explores direct orthogonal collocation optimal control techniques combined with fast geometric path planning algorithms to reduce computation time and increase solution accuracy for SUAS path planning missions. Previous work in the two-dimensional plane demonstrated a methodology to provide optimal flight paths through defined simplex corridors and simplified the optimal control parameter bounds by formulating the problem in the barycentric coordinate system. These methodologies are extended in this paper for three-dimensional flight and solved with two different formulations for flight in an urban environment. The first formulation solves the constrained optimal control problem using a single phase while modeling the building constraints with superquadric functions. The second formulation implements the simplex methodology, eliminating polygonal constraints from the search domain, and solving the optimal path in a multiple phase approach. Results illustrate the benefits gained in computation time and accuracy when implementing simplex methods into the optimal control design and provide a foundation for closing the gap to real-time, onboard operations for unmanned vehicle path planning.

Keywords: Optimal Control; Path Planning; Unmanned Vehicles.

1. Introduction

Small Unmanned Aircraft Systems (SUAS) have been integrated into the mission capabilities of the Department of Defense (DoD) and have been recognized as a critical asset in the force structure as they provide a capability that reduces the risk to human life in dangerous or repetitive missions.[1] By continuing to further incorporate SUAS into mission planning on the battlefield, SUAS systems, sensors, and analytical tasks will be streamlined and human interaction to the mission can be conducted safely.[2] The demand for unmanned aircraft capabilities has become paramount across the DoD and civilian industries. Specifically, Manned Unmanned Teaming (MUM-T) is a capability that allows the manned aircraft to monitor and perform mission objectives while close interactions with complex and contested environments are augmented and enhanced with SUAS to ensure operational success.[3] Considering the urban environment, manned aircraft can experience threats from ground operations that significantly reduce their ability to accomplish the mission. In teaming with SUAS, the manned aircraft can maintain a safe operating distance while unmanned aircraft can expand the operating envelope of the mission through system sensors. This work focuses on transitioning the SUAS from an initial mission location to a secondary mission location which requires travel though a constrained urban environment without interaction or control from the manned aircraft.

Optimal control techniques are used to develop these flight trajectories through highly constrained environments. The common challenges that exist in producing real-time onboard flight trajectories are addressed herein. First, convergence to a solution is not guaranteed and the computation times required to achieve a solution can vary greatly based on the problem setup. Second, the method upon which the constrained environment is modeled and implemented can significantly effect the computation speed and solution convergence. These issues can be attributed to the problem formulation, the implementation of the constraints, and the initial guess (seed) provided to the NLP

*PhD Candidate, Department of Aeronautics and Astronautics, 2950 Hobson Way, Lt. Colonel, USAF.
†Controls Science System Center of Excellence, Wright-Patterson AFB, OH, 45433.

solver. Further, system parameters must be bounded appropriately to ensure the space is adequately searched and results in a feasible solution. This includes bounds on the search domain and mission time which can often be arbitrary values affecting the computation ability of the NLP solver.

To overcome these issues, an expansion of a two-dimensional approach is taken where fast geometric path planning algorithms are used in conjunction with the barycentric coordinate system to solve for optimal paths through a series of simplexes.[4,5] First, the space is discretized into a set of tetrahedrons based on the constrained urban environment and heuristic search algorithms are used to determine efficient simplex corridors from the initial point to the specific terminal location, free of any dynamic vehicle constraints. The proposed solution integrates the simplex search corridor with optimal control algorithms and allows for efficient, feasible, and multi-control solutions through a phased approach in the optimal control solver.

2. Background

Numerical solutions to optimal control problems are often solved using indirect or direct methods. Indirect methods use the calculus of variation to form the Hamiltonian, resulting in a two-point boundary value problem. The optimal solution is determined by solving the first-order optimality conditions while minimizing the Hamiltonian with respect to the control. With this method, a good approximation is required for the states, co-states, control and time. However, the optimality conditions can often be difficult to obtain and determining a realistic estimate of the co-states is not intuitive.

Alternatively, direct methods transcribe the infinite-dimensional optimal control problem into a finite-dimensional optimal control problem with algebraic constraints, also known as a Nonlinear Programming (NLP) problem.[6] Solutions are acquired using orthogonal collocation methods, polynomial approximation of the state, and numerical integration through Gaussian quadrature. The state, x, is approximated at a set of collocation points described as

$$x(\tau) \approx \tilde{x}_N(\tau) = \sum_{i=1}^{n+1} x_i L_i(\tau) \tag{1}$$

where $\tilde{x}_N$ is the N point approximation of $x(\tau)$, x_i represents the weight function, $L_i(\tau)$ is the Lagrange polynomial basis

$$L_i(\tau) = \prod_{\substack{i=0 \\ j \neq i}}^{n} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{2}$$

and τ represents an affine transformation of the time t on the interval from $(-1, 1)$ by

$$\tau = \frac{2t - (t_f + t_0)}{t_f - t_0}. \tag{3}$$

For this research the Legendre-Gauss-Radau points will be implemented.[7] This method is termed global as each collocation point is solved simultaneously rather than other fixed interval methods such as a 3 or 5 point formula method.[8]

One disadvantage of the direct method results from the discretization of the optimal control problem producing several minima, leading to a solution that may be far from the global optimum. To minimize this affect, an accurate prediction of the state, control, and time are required to seed the NLP. The quality of the prediction will have a direct affect on the feasibility of the solution, as there is no guarantee of convergence to a global minima with direct methods. Many algorithms have been proposed previously to produce an initial guess to the solution, including Dubins path algorithms[9] and heuristics[10,11] with computation time and accuracy being the limiting factor for complete hybrid solutions.

Implementing physical building constraints into the path planning optimal control problem has proven to be costly in both accuracy and computation time.[5] Previous work has shown different methods for implementing constraints in the optimal control problem. Constraint functions can be modeled into the problem domain and the optimal control problem can be solved with differentiable path constraint functions such as superquadric ellipsoids, defined as superellipsoids. Superellipsoids have been used in numerous applications to represent three-dimensional shapes, such as modeling city streets or designing aircraft components.[12] Smith used superquadric modeling to represent odd-shaped probability regions for aircraft collision avoidance.[13] This method for modeling constraint functions allows for the designer to shape the edges of the constraint by altering parameters in the base equation cenetered at $(0, 0, 0)$,[14]

$$\left(\left(\frac{x}{a_1} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{a_2} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{a_3} \right)^{\frac{2}{\epsilon_1}} = 1 \tag{4}$$

where constants a_1, a_2, and a_3 set the widths and height of the superellipsoid and ϵ_1 and ϵ_2 vary the cross-section parallel and perpendicular to the x, y plane respectively. Although superellipsoids can begin to represent polygonal shapes as ϵ_i approaches zero, the additional computation time and inconsistencies that result are not practicle for on-board SUAS computations. Alternatively, the research herein examines the effectiveness of using computationally efficient path planning algorithms leveraged from a two-dimensional approach where fast geometric search techniques are used to seed the NLP in solving the optimal control problem to acquire SUAS path trajectories in constrained environments. The proposed method discretizes the search space into a simplex set and eliminates constraints from the defined search corridor containing the path solution. Building upon fast geometric path planning algorithms in two-dimensions,[15] previous work has demonstrated the simplex methodology for optimal control path planning in highly constrained urban environments.[4,16]

Path planning algorithms in three-dimensions have included Dubins path trajectories, A*, and Rapidly exploring Random Tree (RRT) search algorithms.[17] These algorithms have provided geometric path solutions through simple constrained environments, implementing minimum turn radius commands to determine feasible flight paths while constraints are avoided by implementing a series of minimum radius turns, or simply increasing the vehicle turn radius.[18] Although path solutions were attained and the effectiveness of the A* algorithm was demonstrated, these methodologies rely on a geometric path solution without consideration for optimality or rate limited vehicle control parameters such as aircraft angle rates or speed control.

3. Methodology

Several challenges are presented when expanding the methodologies of the two-dimensional problem to three-dimensional space. First, the discretization algorithm used in the two-dimensional space and the resulting Dubins path solution, is not capable of expanding to the third dimension in its current state. To apply the same concepts for discretizing the space, a simple constraint map, shown in Figure 1, is constructed so that a feasibility analysis can be performed and demonstrated.

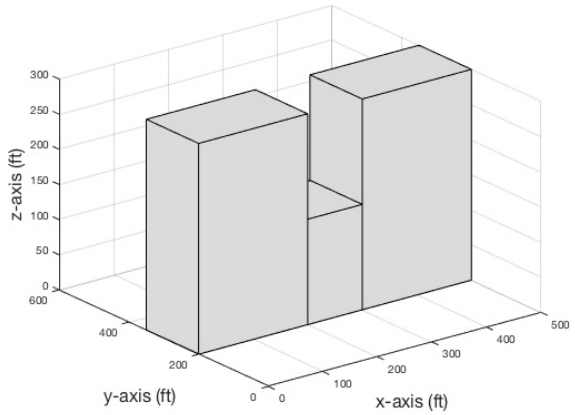

Fig. 1. Three-Dimensional Constraint Map

3.1. *Discretization of the Domain*

Given the coordinates of the constraints and the size of the domain, the space is partitioned laterally in the x-direction and longitudinally in the y-direction based on the length and width of each constraint. The z-axis is partitioned for each independent height level of the constraints. The simple constraints shown in Figure 1 result in two partitions along the x and y-axis and a single partition along the z-axis. For constraints modeled as rectangular prisms, this results in a cubed space. Any cube that contains the same space as a constraint is eliminated from the discretized search space.

In the two-dimensional problem, a Constrained Delaunay Triangulation (CDT) was performed, providing a set of three-sided simplexes. Expanding to three dimensions, four-sided simplexes, or tetrahedrons, are required. To quickly form the simplex discretized space, five tetrahedrons are incorporated into each cube as shown in Figure 2. By this method, two tetrahedrons occupy the top side of the cube, two tetrahedrons occupy the bottom side of the cube, and a single tetrahedron is placed in the center of the cube. Each simplex cube is then populated into the discretized space in a checkerboard-like fashion to assure the simplex edges line up on each connecting cube. This results in a three-dimensional discretized space of four-sided simplexes as shown in Figure 2.

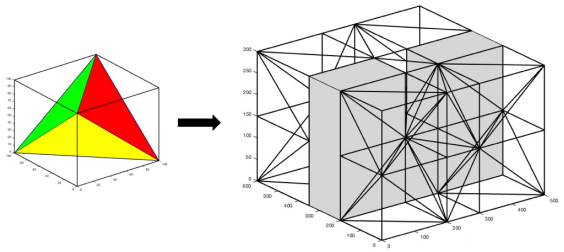

Fig. 2. Three-Dimensional Discretization

With a discretized space defined, a search method is required to find a simplex corridor containing a feasible path solution from the initial starting point to the terminal point. Similar to the two-dimensional approach, an A* algorithm is implemented based on the equation

$$f(n) = g(n) + h(n). \tag{5}$$

Here, $g(n)$ represents the *cost* defined by the Euclidean distance from the mid-point of the current simplex to the mid-point of each connecting simplex. The *cost to go*, $h(n)$, is defined by the Euclidean distance from the midpoint of the connecting simplex to the terminal point of the scenario. The heuristic defined is adequate for this simplified case, however, as the algorithm is developed with a more realistic constraint map, the fidelity of the heuristic implemented should be improved to include flight characteristics of the SUAS. Comparing a Dynamic Programming (DP) search algorithm to an A* approach with the same heuristic, the DP approach results in the complete search of the space. However, due to the simplicity and symmetry of the constraint map, multiple paths of the same minimal cost exist. To eliminate the subjectivity of choosing one of several paths with the same cost, the A* algorithm is chosen to be implemented, where the first completed search corridor identified is accepted.

The A* search algorithm provides a simplex corridor that contains a geometric path solution but may not be feasible due to vehicle dynamics and parameter rate limits. Corrections are made to the resulting corridor to account for cases where the path is extended unnecessarily due to the constrained field. The first case adresses the situation in which the path re-enters a simplex that is already contained in the path solution, resulting in an infinite loop, and

therefore a restriction to enter a simplex already contained in the simplex corridor is enforced. Second, the search corridor avoids a constraint by entering a set of simplexes congruent to the current corridor. In this case, the congruent simplex corridors have three shared edges, extending the corridor length unnecessarily. These congruent corridors are eliminating, resulting in a minimal set. With these corrections applied, a tetrahedron corridor is defined through the three-dimensional discretized space, free of constraints.

3.2. *Algorithm Development*

For each of the formulations posed herein, the optimal control problem is solved with the direct method using the general purpose optimal control solver GPOPS-II. Inputs are required to define the search domain, accomplished with defined bounds on the state, control, and time vectors. These parameter bounds are often problem specific and subjective in nature, further complicating the problem set-up. Additionally, the optimal control problem must be defined by the objective function, the dynamic constraints, the path constraints, and the event constraints. Each of these inputs have an impact to the quality of the solution returned as well as the time required to converge to a solution. Finally an initial guess of the states, control, and time is required to seed the NLP. The following sections describe the methodology used to develop the values and functions for each required input to the optimal control solver using a simplex formulation.

3.2.1. *GPOPS-II Phased Solution*

GPOPS-II is described as a general purpose computational tool for solving multiple-phase optimal control problems using variable-order Gaussian quadrature collocation methods.[19] Each phase is defined with a set of dynamic constraints, path constraints, integral constraints, and parameter constraints. Phases are linked through event constraints that relate information at the start and terminal point of each phase and allow for time, state, and control variables to be continuously transitioned through each phase.[20,21]

For this method, a solution through one simplex can be represented as one phase in GPOPS-II. Formulating the optimal control problem through a simplex set provides the basis for a trajectory solution that traverses through a corridor of simplexes each represented as a single phase, linked together to determine the optimal solution contained inside the defined search space. It is recognized here that the solution found by this method is dependent on the simplex corridor that is presented to the optimal solver. Due to the properties of the A* search algorithm, this corridor may not provide the global optimal solution, rather a local optimal solution may be determined. Therefore, this research focuses on determining feasible solutions that satisfy the dynamic constraints consistently and efficiently as opposed to the necessity of arriving at the global optimal trajectory for every simulation.

The following defines the optimal control problem in terms of a phased approach. Each simplex is represented in barycentric coordinates with appropriate dynamic, path, and parameter constraints. The number of phases required is problem specific and defined after the space has been discretized into a simplex set. The total number of phases in a solution is represented with the variable P.

3.2.2. *Coordinate Transformation*

In order to simplify the optimal control solver input parameters, the problem is expressed in the barycentric coordinate system. Given a tetrahedron shown in Figure 3,

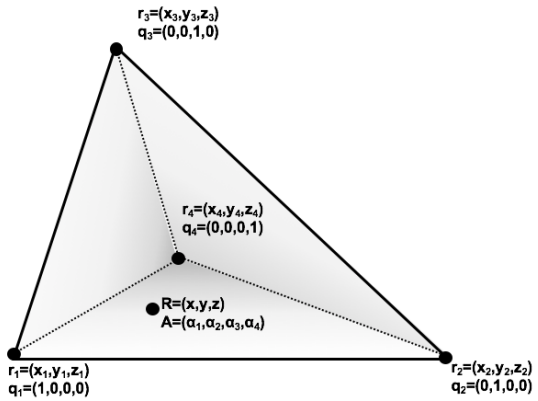

Fig. 3. Barycentric Coordinate Frame for a Tetrahedon

each vertex is defined in Cartesian coordinates as

$$\mathbf{r_i} = (x_i, y_i, z_i) \ \ \forall i \in [1...n] \tag{6}$$

where n is equal to the number of sides of the simplex. The solution in this work is limited to the three-dimensional plane for which $n = 4$. Each point within the simplex can be represented as an ordered quartet of real numbers, representing the weighted distribution to each vertex. Each vertex is defined in barycentric coordinates as

$$\mathbf{q_1} = (\mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \tag{7}$$
$$\mathbf{q_2} = (\mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{0}) \tag{8}$$
$$\mathbf{q_3} = (\mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{0}) \tag{9}$$
$$\mathbf{q_4} = (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{1}) \tag{10}$$

while any point within the simplex is represented with the corresponding weights to each vertex

$$\mathbf{A} = (\alpha_\mathbf{1}, \alpha_\mathbf{2}, \alpha_\mathbf{3}, \alpha_\mathbf{4}). \tag{11}$$

Given a set of barycentric weights, the Cartesian coordinates can be represented as

$$\mathbf{R} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \alpha_\mathbf{i} \mathbf{r_i}. \tag{12}$$

Expanding Eq. 12, the expression below represents the weights of the barycentric coordinate frame in terms of the Cartesian coordinates,

$$\alpha_{\mathbf{i}} = \mathbf{T}^{-1}(\mathbf{R} - \mathbf{q_N}) \ \ \forall i \in [1...N-1] \tag{13}$$

and representing the final weight α_N in terms of the preceding weights, $1 : N-1$,

$$\alpha_N = 1 - \sum_{i=1}^{N-1} \alpha_i \tag{14}$$

where N is defined by the number of vertices in the simplex and $\mathbf{R} - \mathbf{q_N}$ is a $(N-1) \times 1$ vector summation of the Cartesian coordinates.

In three dimensions, $\mathbf{T}$ is a 3×3 matrix defined by

$$\mathbf{T} = \begin{pmatrix} x_1 - x_4 & x_2 - x_4 & x_3 - x_4 \\ y_1 - y_4 & y_2 - y_4 & y_3 - y_4 \\ z_1 - z_4 & z_2 - z_4 & z_3 - z_4 \end{pmatrix}, \tag{15}$$

where x_i, y_i define the vertex locations of each simplex selected such that the points are not collinear.

3.2.3. *SUAS Dynamics*

The dynamics of the aircraft are formulated with a five state model, representing the position of the SUAS in the $x(t)$, $y(t)$, $z(t)$ directions, the heading angle, $\theta(t)$, and the pitch angle, $\psi(t)$. The control for the aircraft is the change in heading angle rate, $u_1(t)$, the change in pitch rate, $u_2(t)$, and the velocity, $v(t)$. The three-dimensional SUAS dynamics are described below as a relationship between the states and the controls[22] for each phase (p):

$$\dot{x}^{(p)}(t) = v(t)cos(\psi(t))cos(\theta(t)) \tag{16}$$
$$\dot{y}^{(p)}(t) = v(t)cos(\psi(t))sin(\theta(t)) \tag{17}$$
$$\dot{z}^{(p)}(t) = v(t)sin(\theta(t)) \tag{18}$$
$$\dot{\theta}^{(p)}(t) = u_1(t) \tag{19}$$
$$\dot{\psi}^{(p)}(t) = u_2(t) \tag{20}$$

$\forall p \in [1...P]$.

3.2.4. *Objective Function*

The objective is to minimize the final time traversing from the initial location to the final location. This is accomplished by advancing through each simplex of the defined corridor in a multi-phased approach with the performance measure defined within each simplex by

$$J^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} dt \ \ \forall p \in [1...P] \tag{21}$$

and the complete objective formed by summing the flight time within each simplex

$$J = \sum_{i=1}^{P} J^{(p)} \tag{22}$$

where t_0 and t_f represent the initial and final time of each phase respectively.

3.2.5. *Dynamic Constraints*

With the barycentric coordinate system and the SUAS dynamics defined, the dynamic constraints of the optimal control problem can now be formulated. The desire is to solve the optimal control problem in a phased approach through a corridor of simplexes, or phases, and therefore the dynamics are represented in terms of the barycentric coordinate frame. Characterizing a tetrahedron, the barycentric coordinates are expressed as a function of the state variables

$$\alpha_i = f_i(x, y, z), \tag{23}$$

further defined as

$$\alpha_1 = \frac{(EI-FH)(x-x_4)-(BI-CH)(y-y_4)+(BF-CE)(z-z_4)}{det(T_3)} \tag{24}$$
$$\alpha_2 = \frac{-(DI-FG)(x-x_4)+(AI-CG)(y-y_4)-(AF-CD)(z-z_4)}{det(T_3)} \tag{25}$$
$$\alpha_3 = \frac{(DH-EG)(x-x_4)-(AH-BG)(y-y_4)+(AE-BD)(z-z_4)}{det(T_3)} \tag{26}$$

where A through I is defined by the mapping

$$\mathbf{T} = \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix} = \begin{pmatrix} x_1 - x_4 & x_2 - x_4 & x_3 - x_4 \\ y_1 - y_4 & y_2 - y_4 & y_3 - y_4 \\ z_1 - z_4 & z_2 - z_4 & z_3 - z_4 \end{pmatrix}. \tag{27}$$

The sum of the weights must equal unity, therefore the fourth coordinate is expressed as

$$\alpha_4 = 1 - \alpha_3 - \alpha_2 - \alpha_1. \tag{28}$$

Evaluating the gradient of f_i defined in Equations 24 through 26 and 28,

$$\dot{\alpha}_i = \nabla f_i \dot{\underline{x}} \tag{29}$$

for

$$\underline{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{30}$$

yields the dynamic equations with respect to the tetrahedron resulting in

$$\dot{\alpha}_1^{(p)} = \frac{(EI-FH)\dot{x}^{(p)}-(BI-CH)\dot{y}^{(p)}+(BF-CE)\dot{z}^{(p)}}{det(T_3)} \tag{31}$$
$$\dot{\alpha}_2^{(p)} = \frac{-(DI-FG)\dot{x}^{(p)}+(AI-CG)\dot{y}^{(p)}-(AF-CD)\dot{z}^{(p)}}{det(T_3)} \tag{32}$$
$$\dot{\alpha}_3^{(p)} = \frac{(DH-EG)\dot{x}^{(p)}-(AH-BG)\dot{y}^{(p)}+(AE-BD)\dot{z}^{(p)}}{det(T_3)} \tag{33}$$
$$\dot{\alpha}_4^{(p)} = -\dot{\alpha}_1^{(p)} - \dot{\alpha}_2^{(p)} - \dot{\alpha}_3^{(p)} \tag{34}$$

$\forall p \in [1...P]$, where the change in the Cartesian x, y, and z positions are defined by the aircraft dynamics in Eq. 16 through 18. The state vector for the three-dimensional problem can now be represented as the barycentric coordinates, heading angle, and pitch angle of the SUAS.

$$X = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \theta, \psi)^T \tag{35}$$

3.2.6. *Path Constraints*

Inequality path constraints represent hard constraints in the optimal control problem that cannot be violated. These constraints are formulated as

$$c_{min}^{(p)} \leq c^{p}(y^{(p)}, u^{(p)}, t^{(p)}) \leq c_{max}^{(p)} \;\; \forall p \in [1...P] \tag{36}$$

where c is a function of the state, control, and time that influences the vehicle's trajectory. Often, these path constraints represent no-fly zones, terrain, or buildings. For comparison, this work will evaluate two formulations. The first will implement building constraints into the optimal control problem with differentiable functions. The second will eliminate the requirement to define building constraints by removing them from the search field through a defined simplex corridor.

3.2.7. *Event Constraints*

Event constraints are implemented to maintain a continuous transition of the state, control, and time variables between each phase. Referencing the two-dimensional case, this condition requires that each vertex of a simplex be defined as the q_1, q_2, or q_3 vertex. As the path trajectory traverses across an edge, one of the weights (states 1-3) will be zero as the weight associated with the opposing vertex has no contribution to the location of the point. As the new phase begins, it is imperative that the states of the next simplex match the states of the previous simplex. In other words, the opposite vertex of the new simplex must accept the zero value and the associated weights for the other vertices must match appropriately in the state vector.[16] This is illustrated in Figure 4.

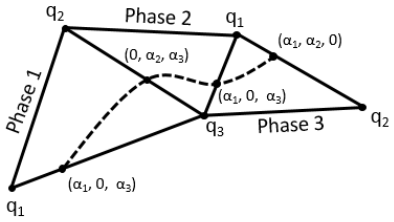

Fig. 4. Event constraints through notional simplex corridor

Extending this methodology to the third dimension, the path will transition from one simplex to the next through a shared face. The three shared vertices on this face can take a barycentric weight value between zero and one. However, the fourth vertex on each simplex must accept a zero value as it provides no contribution to the path position when located at a simplex transition boundary. Implementing these bounds within the optimal control solver increases the computation speed by providing a required direction for the search.

3.2.8. *Bounds*

The search space of the optimal solver is limited by the bounds applied to the state parameters, the control, and the mission time. Often, these bounds can be difficult to determine. If they are set too large, convergence times can become excessive. If they are set too small, there is a greater probability the solution will converge to a local minimum instead of continuing to search for the global minimum. Additionally, if the bounds are set too restrictive, the optimal solution may no longer be in the search space and a feasible solution may not be found.

By defining the problem with a simplex set in the barycentric coordinate system and solving each simplex as a separate phase, the bounds become simplified and strictly defined for each problem. The weights of the barycentric coordinates are defined from 0 to 1, therefore the bounds on the position states become

$$0 \leq \alpha_i \leq 1 \;\; i = 1, 2, ...N, \tag{37}$$

where N defines the number of vertices in the simplex. The time vector is bounded with an upper and lower limit in each phase. The furthest distance the SUAS can travel through any simplex is equal to the length of the longest edge. The bounds on the remaining state and control parameters are specific to vehicle characteristics and are set such that a tractable scenario can be accomplished.

4. Optimal Control Problem

Each tetrahedron in the defined search corridor is solved as a single phase in the optimal control solver and each phase is connected through event constraints. The dynamics constrain the path through each tetrahedron in barycentric coordinates. The optimal control problem formulation has been consolidated as follows.

Minimize the cost functional

$$J = \sum_{p=1}^{P} J^{(p)} \tag{38}$$

where

$$J^{(p)} = \int_{t_0^{(p)}}^{t_f^{(p)}} dt \;\; \forall p \in [1...P] \tag{39}$$

subject to the dynamic constraints in Eqs. 31 to 34 and Eqs. 19 to 20, and the vehicle dynamics defined in Eqs. 16 to 18. The control is placed on rate of change of the pitch angle, heading angle, and velocity

$$u_1^{(p)}(t) = \dot{\psi}^{(p)}(t) \tag{40}$$

$$u_2^{(p)}(t) = \dot{\theta}^{(p)}(t) \tag{41}$$

$$u_3^{(p)}(t) = v \tag{42}$$

with the state vector defined in Eq. 35. The boundary conditions are given as the initial and final constraints on the

states,

$$\mathbf{X}^{(1)}(t_0^{(1)}) = ((\alpha_1)_0, (\alpha_2)_0, (\alpha_3)_0, (\alpha_4)_0, (\theta)_0, (\psi)_0) \quad (43)$$

$$\mathbf{X}^{(P)}(t_f^{(P)}) = ((\alpha_1)_f, (\alpha_2)_f, (\alpha_3)_f, (\alpha_4)_f, (\theta)_f, (\psi)_f) \quad (44)$$

For the scenario herein, inequality path constraints representing bounds on the state, control and time are defined as

$$0 \leq \alpha_1^{(p)}, \alpha_2^{(p)}, \alpha_3^{(p)}, \alpha_4^{(p)} \leq 1 \quad (45)$$

$$|\theta^{(p)}| \leq 180 \ \ deg \quad (46)$$

$$|\dot{\theta}^{(p)}| \leq 25 \ \ deg/s \quad (47)$$

$$|\psi^{(p)}| \leq 30 \ \ deg \quad (48)$$

$$|\dot{\psi}^{(p)}| \leq 10 \ \ deg/s \quad (49)$$

$$10ft/s \leq v^{(p)}(t) \leq 30ft/s \quad (50)$$

$$0 \leq t^{(p)} \leq \rho \frac{edge_{max}^{(p)}}{v} \quad (51)$$

where $edge_{max}$ defines the longest edge of a simplex and ρ is a scaling factor to allow for extended time in a simplex in the presence of wind or other exogenous inputs. Finally, event constraints are included to maintain a continuous transition of the state variables between each phase,

$$X_o^{(p+1)} - X_f^{(p-1)} = 0 \ \ \forall p \in [2...P]. \quad (52)$$

The key GPOPS-II parameters used herein are listed below in Table 1.

GPOPS-II User Settings	
Mesh Method	hp-PattersonRao
Mesh Tolerance	10^{-3}
NLP Solver	IPOPT
Derivative Supplier	AdiGator
Method	RPM-differential
NLP Tolerance	10^{-5}
Min Collocation Points	4
Max Collocation Points	10
Mesh Fraction	$\frac{1}{2}$*ones(1,2)
Mesh Collocation Points	4*ones(1,4)

5. Scenario

A simple three constraint model was developed to analyze the three-dimensional scenario. In order to illustrate the effectiveness of the simplex approach, the scenario is solved with previous methods in the literature in a single phase using superellipsoid constraint functions and compared to the solution using a simplex discretization. For the simplex method, a discretization of the space is performed and an A* search algorithm determines the optimal search corridor. An initial guess for the path solution is determined by connecting the centroid of each simplex through the search corridor. A two-point finite differencing scheme is implemented to acquire initial vectors for the heading and pitch angle, while the heading rate and pitch rate are initiated with the zero vector and the velocity is presented at a maximum value. In order to better compare the two solutions in each formulation, this initial path solution was used to seed the NLP in both the single phase and multiple phase scenarios. The building constraints consists of three connected polygons, representing a series of buildings along a street with the middle constraint only half the height of the other two as shown in Figure 5.

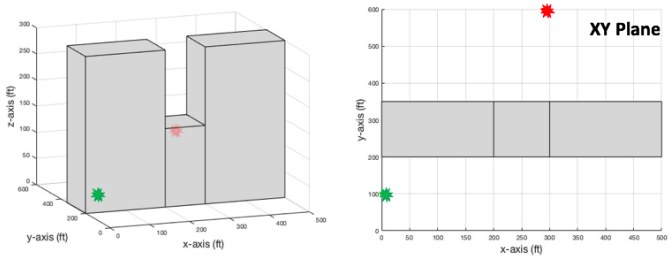

Fig. 5. Three-Dimensional Constraint Map

The aircraft begins at level flight flying parallel to the building constraints and is required to climb over the center building and descend to a terminal point perpendicular to the original path. The initial and final aircraft constraints are defined as

$$(x_0, y_0, z_0, \theta_0, \psi_0) = (0, 100, 50, 0, 0) \quad (53)$$

$$(x_f, y_f, z_f, \theta_f, \psi_f) = (290, 600, 50, \tfrac{\pi}{2}, 0), \quad (54)$$

where the initial and terminal location are defined by a green and red asterisk respectfully, as shown in Figure 5.

5.1. *Single Phase Formulation*

The first formulation illustrates a single phase solution. The problem is solved using Cartesian coordinates with superellipsoid constraint functions implemented in the optimal control solver to model each of the three buildings. The dynamics consist of a five state model consisting of the Cartesian coordinates expressed in Eq. 16 to 18, the heading angle, θ, and the pitch angle, ψ. The control is implemented on the change in heading angle, $\dot{\theta}$, the change in pitch angle, $\dot{\psi}$, and the velocity. The bounds on each of the first three states are consistent with the search domain space and are defined as

$$0 \leq x \leq 500ft \quad (55)$$

$$0 \leq y \leq 600ft \quad (56)$$

$$0 \leq z \leq 300ft. \quad (57)$$

The bounds on the remaining states and control parameters are consistent with those defined in Eq. 46 to 51.

The three building constraints are modeled with a superellipsoid function and designed as an inequality path constraint defined as

$$\left(\left(\frac{x_a - x_c}{a_1} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y_a - y_c}{a_2} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z_a - z_c}{a_3} \right)^{\frac{2}{\epsilon_1}} \leq 1. \quad (58)$$

Here, the a subscript on the Cartesian coordinates refers to the aircraft position, the c subscript defines the center

point of the superellipsoid, while the principle axes in each direction is defined by a_i. The curvature at the edges of the superellipsoid is defined with the ϵ_1 and ϵ_2 term which represent cuboids when they take on values less than one and greater than zero.[14] For this work, both ϵ_1 and ϵ_2 were set to 0.01. Figure 6 shows the superellipsoid shape.

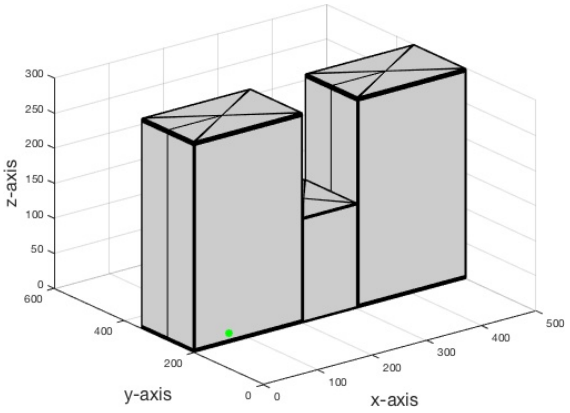

Fig. 6. Three-Dimensional Superellipsoid Constraint Map

Although the constraint shape looks polygonal, the edges are rounded ever so slightly, creating a small error between the polygonal shape and the superellipsoid. This error is characterized in the results. Finally, the natural log is taken on both sides of Eq 58 to minimize the impact the large constraint values can have on the computation time of the NLP solver.

With the optimal control problem defined for a single phase, an optimal flight path can be computed. Results for the state and control parameters are compared to the results for a multi-simplex solution and are shown in the subsequent sections.

5.2. *Simplex Formulation*

The new proposed formulation demonstrates the simplex solution and is solved in barycentric coordinates with a phased approach in the optimal control solver. The space is discretized into a tetrahedronal set and an A* search algorithm is implemented based on a mid-point heuristic to determine the search corridor as described in Section 3.1. The defined simplex corridor is shown in Figure 7.

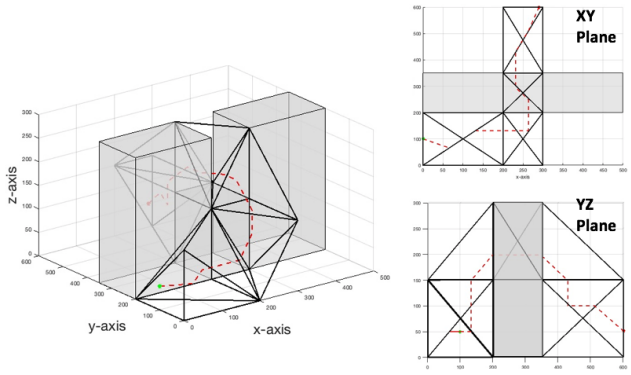

Fig. 7. Three-Dimensional Simplex Search Corridor

The dashed red line indicates a path solution, connecting the mid-point of each simplex of the search corridor and is used to seed the NLP. Given this defined search corridor, the optimal path is computed with a phased approach as defined in the optimal control problem in Section 4. These results are compared to the single phase solution in computational time and accuracy.

6. Results

In order to draw comparisons between the two formulations, the same path solution was used to seed the NLP in the single phase and multi-phase formulation. This initial path solution can be seen in Figure 8 with the dashed red line. The green asterisks reflect the first formulation in which the optimal solution is solved in one phase within the global search domain of the space and satisfies the superellipsoid constraint function. The blue asterisks reflect the second formulation where the optimal solution is determined through each simplex of the defined search corridor.

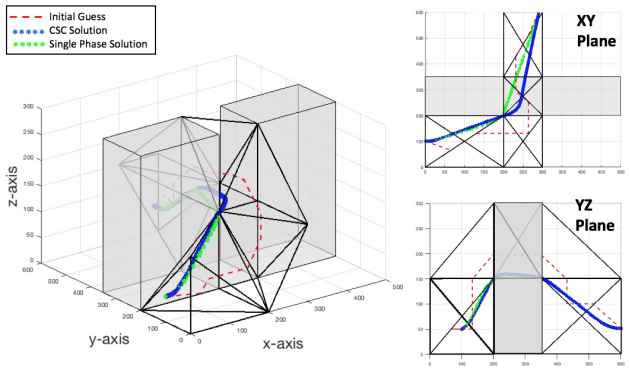

Fig. 8. Three-Dimensional Optimal Control Path Solution

Variation between the two paths can be seen as they climb over the center building with the single phase solution taking a more direct route to the terminal point. The

angular state and control parameters are shown in Figure 9.

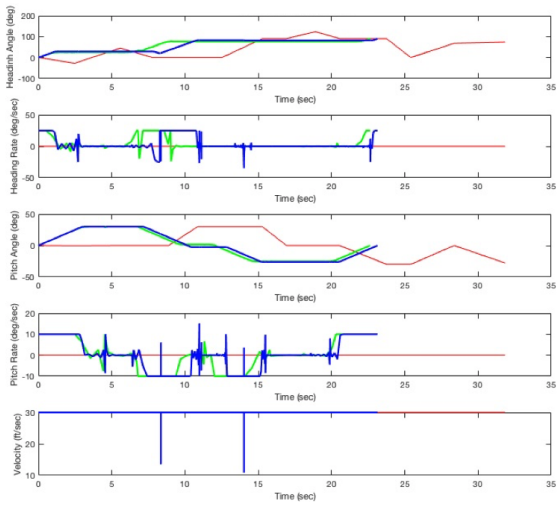

Fig. 9. Three-Dimensional Optimal State & Control

The top plot in Figure 9 shows the heading angle. As expected with larger tetrahedrons, the centroid solution used to seed the NLP has the most variation in the heading and takes the longest time to complete the path. The single phase and simplex solution only differ in the timing of the turn to fly over the center constraint. The third plot defines the pitch angle with all three path solutions resembling the same angular requirements, however, the computation time of the centroid solution is significantly longer. The second and fourth plots describe the angle rates for the heading and pitch respectively and resemble Pontryagin's principles as expected given the rate limitations on the parameters. Finally, the fifth plot shows the SUAS maintains max speed throughout the simulation in each scenario, however this would not be the case when tighter turn radii are required.[4]

The computation and objective times are shown in Table 2 for each of the two solutions.

Solution Type	Comp Time (s)	Obj Time (s)
Single Phase	45.47	22.631
Simplex Solution	5.21	22.77

The objective times between the two scenarios only differ by less than two-tenths of a second while the difference in computation time is significant at over 40 seconds. The disparity in the objective time can be explained in the difference of the constraint function models. Superquadrics were used in modeling the constraints in the single phase solution. The small ϵ_i values increase the sharpness of the constraint edges, but also increases the exponential power of the constraint function. This creates large gradient values within the NLP resulting in a significant increase in computation time. With these epsilon values implemented, there remains a small error in the shape of the superquadric when compared to the polygonal constraint used in the simplex solution. Characterizing this error is accomplished by reviewing Barr's work for modeling the volume of a superellipsoid,[23] defined in his work as

$$V_E = \frac{2}{3} a_1 a_2 a_3 \epsilon_1 \epsilon_2 \beta\left(\frac{\epsilon_1}{2}, \frac{\epsilon_1}{2}\right) \beta\left(\epsilon_2, \frac{\epsilon_2}{2}\right) \tag{59}$$

where β represents the beta function and the a_i terms represent the principle axes of the superellipsoid in each direction. The difference between the polygon shape constraint and the superquadric is $1099.6 ft^3$, which results in a 0.012% error. Although this error is small, it is concentrated at the edges of the constraint. The rounded edge of the superquadric function as well as the spacing of the collocation points allows for a more direct flight path over the center constraint. However, the path violates the constraint modeled as a polygon as the path skips over the corner of the constrained edge. The SUAS path solution over the first edge of the center constraint can be seen in the left side and top image in Figure 10.

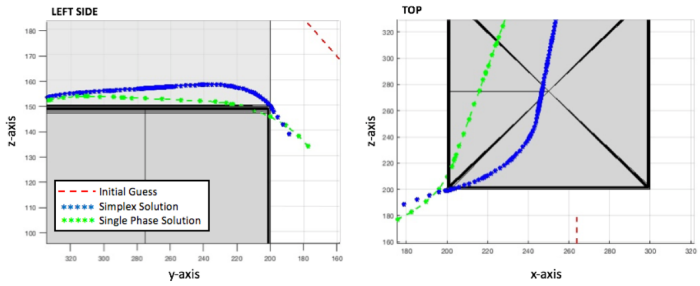

Fig. 10. Three-Dimensional Path and Constraint Comparison

The superquadric model is a good representation of a polygonal constraint, however, to guarantee the true polygonal constraint is not violated, a subjective safety buffer would have to be included into the superquadric model to assure a feasible flight path. By determining a path through a defined simplex corridor, the constraints are eliminated from the domain of the problem and a feasible flight path is presented based on strictly defined system parameters.

7. Conclusions

The significant difference between the two and three-dimensional approach for flight through an urban environment is the lack of a readily available three-dimensional discretization and geometric path planning solver. Extending the simplex discretization to three-dimensions, while maintaining the characteristics of a constrained triangulation, is a current research challenge. This work provided a methodology to discretize the constrained domain using simple rectangular prism shapes. Additionally, fast geometric path planners used in the two-dimensional analysis contain A* search algorithm as well as a funnel algorithm based on a heuristic that has been tuned and developed for the specified path solution. The approach taken in this

work implements a straightforward A* search with a centroid heuristic to attain a rudimentary path solution. Both the discretization and initial path generation can be significantly improved to allow for more complex constraint environments and a more accurate path solution to seed the NLP of the optimal control solver.

Given a discretization of the space and an initial path solution, extending the principles of the two-dimensional optimal control problem to three-dimensions is straightforward under the simplex construct. Three-dimensional vehicle dynamics can be implemented with barycentric coordinates and a path solution can be attained by solving the path one simplex at a time and connecting each simplex solution through path constraints in the optimal solver. The results attained in this work illustrate the benefits that can be achieved by formulating the problem with a simplex approach resulting in strictly defined parameter bounds where the constraints imposed by the urban infrastructure are eliminated from the search space of the NLP solver. Future work developing efficient three-dimension discretization and geometric search algorithms will further increase computational speed and accuracy and allow for rapid solutions to more realistic scenarios.

Acknowledgments

The authors would like to thank the Air Force Research Laboratory, Aerospace Systems Directorate, Power and Controls Division, who sponsored the work and provided the challenging problem concept, background information, and continual support.

References

[1] *Joint Concept of Operations for Unmanned Aircraft Systems* (US Department of Defense, Washington, D.C., 2011).

[2] US Air Force Office of Remotely Piloted Aircraft (RPA) Capabilities, Small UAS Flight Plan 2016 - 2036, tech. rep., Deputy Cheif of Staff for Intelligenc, Survaillance, and Reconnaissance (ISR) (2016).

[3] Air Force Research Laboratory, Air Force Research Laboratory Autonomy Science and Technology Strategy, tech. rep., Air Force Research Laboratory, Wright-Patterson AFB, OH (2013).

[4] M. D. Zollars, R. G. Cobb and D. J. Grymin, Simplex Solutions for Optimal Control Flight Paths in Urban Environments, *Journal of Aeronautics and Aerospace Engineering* **6**(3) (2017) p. 8.

[5] M. D. Zollars and R. G. Cobb, Simplex Methods for Optimal Control of Unmanned Aircraft Flight Trajectories, *ASME Dynamics Systems and Controls Conference*, Tysons Corner, VA (2017), p. 10.

[6] G. Huntington, Advancement and analysis of a Gauss pseudospectral transcription for optimal control problems, dissertation, Massachusetts Institute of Technology (2007), p. 207.

[7] D. Garg, Advances In Global Psuedospectral Methods For Optimal Control, dissertation, University of Florida (2011), p. 238.

[8] L. N. Trefethen, *Spectral Methods in Matlab* (Society for Industrial and Applied Mathematics, 2000).

[9] M. Zollars, P. Blue and B. Burns, Wind Corrected Flight Path Planning for Autonomous Micro Air Vehicles Utilizing Optimization Techniques, *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, Hilton Head, SC (2007).

[10] C. J. Humphreys and R. G. Cobb, A Hybrid Optimization Technique Applied to the Intermediate-Target Optimal Control Problem, *Global Journal of Technology and Optimization* **7**(2) (2016).

[11] D. Ferguson, M. Likhachev and A. Stentz, A guide to heuristic-based path planning, *Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS)*, (2005), pp. 1 – 10.

[12] F. Jaklic, Ales; Leonardo, Ales; Solina, *Segmentation and Recovery of Superquadrics* (Kluwer Academic Publishers, Dordrecht, The Netherlands, 2013).

[13] N. E. Smith, Optimal Collision Avoidance Trajectories for Unmanned/Remotely Piloted Aircraft, dissertation, Air Force Institute of Technology (2014), p. 240.

[14] A. H. Barr, Superquadrics and Angle-Preserving Transformations., *IEEE Computer Graphics and Applications* **1**(1) (1981) 11–23.

[15] M. Kallmann, Shortest Paths with Arbitrary Clearance from Navigation Meshes, *Proceedings of the Eurographics SIGGRAPH Symposium on Computer Animation SCA*, (2010).

[16] M. D. Zollars, R. G. Cobb and D. J. Grymin, Simplex Optimal Control Methods for Urban Environment Path Planning, *AIAA Sci-Tech Information Systems Conference*, Orlando, FL (2018), p. 16.

[17] C. Zammit and E.-J. Van Kampen, Comparison between A* and RRT Algorithms for UAV Path Planning, *2018 AIAA Guidance, Navigation, and Control Conference*, (January) (2018).

[18] N. K. Singh and S. Hota, Waypoint Following for Fixed-Wing MAVs in 3D Space, *2018 AIAA Guidance, Navigation, and Control Conference*, (January) (2018).

[19] M. A. Patterson and A. V. Rao, GPOPS II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hpAdaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming, *ACM Transactions on Mathematical Software* **41**(1) (2010) 1–39.

[20] J. T. Betts, Sparse Jacobian updates in the collocation method for optimal control problems, *Journal of Guidance, Control, and Dynamics* **13**(3) (1990) 409–415.

[21] J. T. Betts, Practical methods for optimal control and estimation using nonlinear programming, *Advances in design and control* (2008) p. 434.

[22] D. Hull, *Fundamentals of Airplane Flight Mechanics* (Springer, Berlin, Germany, 2007).

[23] A. H. Barr, Rigid Physically Based Superquadrics, *Graphic Gems III, Chapter III.8*, 1992, pp. 137–159.

Bibliography

1. *Joint Concept of Operations for Unmanned Aircraft Systems*. US Department of Defense, Washington, D.C., 2011.

2. James A Winnefeld and Frank Kendall. Unmanned Systems Integrated Roadmap FY2013-2038. Technical report, US Department of Defense, Washington, D.C., 2013.

3. Elizabeth Bone and Christopher Bolkcom. Unmanned Aerial Vehicles: Background and Issues for Congress. Technical report, 2003.

4. US Air Force Office of Remotely Piloted Aircraft (RPA) Capabilities. Small UAS Flight Plan 2016 - 2036. Technical report, Deputy Cheif of Staff for Intelligenc, Survaillance, and Reconnaissance (ISR), 2016.

5. Air Force Research Laboratory. Air Force Research Laboratory Autonomy Science and Technology Strategy. Technical report, Air Force Research Laboratory, Wright-Patterson AFB, OH, 2013.

6. Jon Harper. Special Operations Gunships to Be Equipped With Improved Sensors. *National Defense*, pages 1–4, 2016.

7. Peter T. Heidlauf. *Optimal UAV Path Planning with Dynamic No-Fly-Zones for Target Geolocation using Line-of-Bearing Measurements and Kalman Filtering*. Ms thesis, Air Force Institute of Technology, 2017.

8. Marc Sands. Lecture Notes, ASYS 625, Non-Linear Control; Air Force Institute of Technology. 2017.

9. Jon Strizzi, I Michael Ross, and Fariba Fahroo. Towards Real-Time Computation of Optimal Controls for Nonlinear Systems. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (August), 2002.

10. UAS Task Force: Airspace Integration Integrated Product Team. Unmanned Aircraft System Airspace Integration Plan. Technical Report March, 2011.

11. RP Optimization Research LLC. GPOPS-II: Next-Generation Optimal Control Software. 2016.

12. Nathan E. Smith. *Optimal Collision Avoidance Trajectories for Unmanned/Remotely Piloted Aircraft*. Dissertation, Air Force Institute of Technology, 2014.

13. Clay J. Humphreys, Richard G. Cobb, David R. Jacques, and Jonah A. Reeger. Dynamic Re-plan of the Loyal Wingman Optimal Control Problem in a Changing Mission Environment. In *AIAA Infotech @ Aerospace*, number January, pages 1–13, 2017.

14. Tadeusz J. Masternak. *Multi-Objective Trajectory Optimization of a Hypersonic Reconnaissance Vehicle with Temperature Constraints*. Dissertation, Air Force Institute of Technology, 2014.

15. Ryan W. Carr and Richard Cobb. An Energy Based Objective for Solving an Optimal Missile Evasion Problem. In *AIAA Guidance, Navigation, and Control Conference*, pages 1–18, 2017.

16. Ryan W. Carr and Ernest Lagimoniere. A Range Safety Footprint Analysis for the Dream Chaser Engineering Test Article Using Trajectory Optimization. *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013.

17. Anil V. Rao, Michael A. Patterson, and William W. Hager. A ph Collocation Scheme for Optimal Control. *Optimal Control Applications and Methods*, pages 1–40, 2013.

18. Angela W Suplisson. *Optimal Recovery Trajectories for Automatic Ground Collision Avoidance Systems (Auto GCAS)*. Dissertation, Air Force Institute of Technology, 2015.

19. Divya Garg. *Advances In Global Psuedospectral Methods For Optimal Control*. Dissertation, University of Florida, 2011.

20. Donald E. Kirk. *Optimal Control Theory: An Introduction*. Dover Publications, 2004.

21. Richard G Cobb. Lecture Slides, ASYS622, Optimal Control; Air Force Institute of Technology. Technical report, Air Force Institute of Technology, 2016.

22. D.A. Benson, G.T. Huntington, T.P. Thorvaldsen, and A.V. Rao. Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method. *Journal of Guidance, Control, and Dynamics*, 29(6):1435–1440, 2006.

23. G.T. Huntington. *Advancement and analysis of a Gauss pseudospectral transcription for optimal control problems*. Dissertation, Massachusetts Institute of Technology, 2007.

24. James K. Miller, Pedro J. Llanos, and Gerald R. Hintz. Optimal Control Framework for Impulsive Missile Interception Guidance. In *Guidance, Navigation, and Control Conference*, number 8, pages 1–14, Boston, MA, 2014.

25. John T. Betts. Practical methods for optimal control and estimation using nonlinear programming. *Advances in design and control*, page 434, 2008.

26. Richard L. Burden and John Douglas Faires. *Numerical Analysis*. 2010.

27. Lloyd N. Trefethen. *Spectral Methods in Matlab*. Society for Industrial and Applied Mathematics, 2000.

28. Michael Zollars, Paul Blue, and Brian Burns. Wind Corrected Flight Path Planning for Autonomous Micro Air Vehicles Utilizing Optimization Techniques. In *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, Hilton Head, SC, 2007.

29. Clay J. Humphreys and Richard G. Cobb. A Hybrid Optimization Technique Applied to the Intermediate-Target Optimal Control Problem. *Global Journal of Technology and Optimization*, 7(2), 2016.

30. Dave Ferguson, Maxim Likhachev, and Anthony Stentz. A guide to heuristic-based path planning. In *Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS)*, pages 1 – 10, 2005.

31. Nidal M. Jodeh. *Optimal UAS Assignments and Trajectories for Persistent Surveillance and Data Collection from a Wireless Sensor Network*. Dissertation, Air Force Institute of Technology, 2015.

32. Franc Jaklic, Ales; Leonardo, Ales; Solina. *Segmentation and Recovery of Superquadrics*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2013.

33. L. R. Lewis. *Rapid motion planning and autonomous obstacle avoidance for unmanned vehicles*. Ms thesis, Naval Postgraduate School, 2006.

34. M.A. Hurni. An information-centric approach to autonomous trajectory planning utilizing optimal control techniques. page 296, 2009.

35. Krithika Mohan. Optimal Path Planning and Trajectory Optimization for. *University of Florida*, pages 1–85, 2011.

36. Clay J. Humphreys, Richard G. Cobb, David R. Jacques, and Jonah A. Reeger. Dynamic Re-plan of the Loyal Wingman Optimal Control Problem in a Changing Mission Environment. In *AIAA Sci-Tech*, pages 1–15, 2016.

37. Alan H. Barr. Superquadrics and Angle-Preserving Transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.

38. Moshe Shimrat. Position of Point Relative to Polygon. *Communications of the ACM*, 5(8):434, aug 1962.

39. Richard Hacker. Certification of Algorithm 112. *Communications of the ACM*, 5(12):606, dec 1962.

40. A V Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

41. Bruce A. Conway. A Survey of Methods Available for the Numerical Optimization of Continuous Dynamic Systems. *Journal of Optimization Theory and Applications*, 152(2):271–306, 2011.

42. Mubbasir Kapadia and Norman I. Badler. Navigation and steering for autonomous virtual humans. *Wiley Interdisciplinary Reviews: Cognitive Science*, 4(3):263–272, 2013.

43. Mubbasir Kallmann, Marcelo; Kapadia. *Geometric and Discrete Path Planning for Interactive Virtual Worlds*. Morgan & Claypool Publishers series, 2016.

44. Alain Juarez-Perez and Marcelo Kallmann. Full-body behavioral path planning in cluttered environments. *Proceedings of the 9th International Conference on Motion in Games - MIG '16*, pages 107–112, 2016.

45. M Kallmann. Dynamic and Robust Local Clearance Triangulations. *ACM Transactions on Graphics*, 33(5):17, 2014.

46. L.E. Kavraki and P. Svestka. Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566 – 580, 1996.

47. Roland Geraerts. Planning Short Paths with Clearance using Explicit Corridors. pages 1997–2004, 2010.

48. Priyadarshi Bhattacharya and Marina L. Gavrilova. Voronoi diagram in optimal path planning. *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*, 1(c):38–47, 2007.

49. Marcelo Kallmann. Navigation Queries from Triangular Meshes. Technical report, 2010.

50. D. F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, Vol. 24(No. 2):167–172, 1998.

51. Sw Sloan. A fast algorithm for constructing Delaunay triangulations in the plane. *Advances in Engineering Software*, 9(1):34–55, 1987.

52. Gerald Farin. Surfaces over Dirichlet tessellations. *Computer Aided Geometric Design*, 7(1-4):281–292, 1990.

53. Paul L. Chew. Constrained Delaunay Triangulations. In *Proceedings of the third annual symposium on Computational Geometry*, pages 215–222, Waterloo, Ontario, 1987.

54. R. Simpson. Locally Equiangular triangulations. *The Coimputer Journal*, 21(3):243–245, 1978.

55. Douglas Demyen and Michael Buro. Efficient Triangulation-Based Pathfinding. In *Proceedings of the 21st National Conference on Artificial Intelligence, AAAI*, pages 942–947, Menlo Park, 2006. AAAI Press.

56. Moshe Sniedovich. Dijkstra's algorithm revisited: The dynamic programming connexion. *Control and Cybernetics*, 35(3):599–620, 2006.

57. Eric V. Denardo. *Dynamic Programming; Models and Applications*. Dover Publications, Mineola, NY, 2013.

58. D. Myers, R. Batta, and M. Karwan. A real-time network approach for including obstacles and flight dynamics in UAV route planning. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 13(3):291–306, 2016.

59. Pearl Dechter, Rina; Judea. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, 32(3):505–536, 1985.

60. Magnus Lie Hetland. *Python Algorithms: Mastering Basic Algorithms in the Python Language*. Apress, 2014.

61. S. M. Lavalle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. *Techreport*, 11, 1998.

62. Emilio Frazzoli. Real-Time Motion Planning for Agile Autonomous Vehicles. 25(1), 2002.

63. Georges Salim Aoude and Jonathan P How. *Two-Stage Path Planning Approach for Designing Multiple Spacecraft Reconfiguration Maneuvers and Application to SPHERES onboard ISS*. Thesis, Massachusetts Institute of Technology, 2005.

64. Georges S. Aoude, Jonathan P. How, and Ian M. Garcia. Two-stage path planning approach for solving multiple spacecraft reconfiguration maneuvers. *The Journal of the Astronautical Sciences*, 56(4):515–544, 2008.

65. Marcelo Kallmann. Shortest Paths with Arbitrary Clearance from Navigation Meshes. In *Proceedings of the Eurographics SIGGRAPH Symposium on Computer Animation SCA*, 2010.

66. Marcelo Kallmann and Mubbasir Kapadia. Navigation meshes and real-time dynamic planning for virtual worlds. *ACM SIGGRAPH 2014 Courses*, pages 1–81, 2014.

67. D.T. Lee and F.P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. In *Networks 14*, pages 393–410, 1984.

68. B Chazelle. A theorem on polygon cutting with applications. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 339–349, 1982.

69. John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry: Theory and Applications*, 4(2):63–97, 1993.

70. Marcelo Kallmann. Flexible and Efficient Navigation Meshes for Virtual Worlds Flexible and Efficient Navigation Meshes. In *Simulating Heterogenous Crowds iwth Interactive Behaviors*. 2016.

71. Jianqiu Xu and Ralf Hartmut Güting. Querying visible points in large obstructed space. *GeoInformatica*, 19(3):435–461, 2015.

72. Ning Brass, Peter; Vigan, Ivo; Xu. Shortest path planning or a tethered robot. *Computational Geometry*, 48(9):732–742, 2015.

73. Melissa M. Tanner, Joel W. Burdick, and Issa A D Nesnas. Online motion planning for tethered robots in extreme terrain. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5557–5564, 2013.

74. Joe Warren, Scott Schaefer, Anil N. Hirani, and Mathieu Desbrun. Barycentric coordinates for convex sets. *Advances in Computational Mathematics*, 27(3):319–338, 2007.

75. Max Schindler and Evan Chen. Barycentric Coordinates in Olympiad Geometry. pages 1–40, 2012.

76. Tim Visser, Coen C. De Visser, and Erik-Jan Van Kampen. Quadrotor System Identification Using the Multivariate Multiplex B-Spline. In *AIAA Atmospheric Flight Mechanics Conference*, number January, pages 1–13, 2015.

77. Mark Meyer, Alan Barr, Haeyoung Lee, and Mathieu Desbrun. Generalized Barycentric Coordinates on Irregular Polygons. *Journal of Graphics Tools*, 7(1):13–22, 2002.

78. Michael S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

79. Michael A. Patterson and Anil V. Rao. GPOPS II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hpAdaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming. *ACM Transactions on Mathematical Software*, 41(1):1–39, 2010.

80. Michael A. Patterson and Anil V. Rao. GPOPS-II manul: A General-Purpose MATLAB Software for Solving Multiple-Phase Optimal Control Problems Version 2 . 1. (October):1–72, 2015.

81. John T. Betts. Sparse Jacobian updates in the collocation method for optimal control problems. *Journal of Guidance, Control, and Dynamics*, 13(3):409–415, 1990.

82. D.G. Hull. *Fundamentals of Airplane Flight Mechanics*. Springer, Berlin, Germany, 2007.

83. Michael D. Zollars, Richard G. Cobb, and David J. Grymin. Simplex Solutions for Optimal Control Flight Paths in Urban Environments. *Journal of Aeronautics and Aerospace Engineering*, 6(3):8, 2017.

84. Alan H. Barr. Rigid Physically Based Superquadrics. In *Graphic Gems III, Chapter III.8*, chapter III-88, pages 137–159. 1992.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
07–12–2018	Doctoral Dissertation	Sept 2015 — Sep 2018

4. TITLE AND SUBTITLE

SIMPLEX CONTROL METHODS FOR ROBUST CONVERGENCE OF SMALL UNMANNED AIRCRAFT FLIGHT TRAJECTORIES IN THE CONSTRAINED URBAN ENVIRONMENT

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Michael D. Zollars, Lt Col, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENY-DS-18-S-078

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory, Aerospace Systems Directorate (AFRL/RQQC)
Attn: Amy Burns
2210 8th Street
WPAFB OH 45433-7542
DSN 674-6542, COMM 937-904-6542
Email: amy.burns.3@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RQQA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States

14. ABSTRACT

In this research, a new hybrid approach is implemented to acquire feasible path solutions for SUAS through highly constrained spaces, investigating narrow corridors, visiting multiple waypoints, and minimizing incursions to keep-out regions. These issues are addressed herein with a new approach by triangulating the search space to define a polygonal search corridor free of constraints while alleviating the dependency of problem specific parameters by translating the problem to barycentric coordinates. Within this Connected Simplex Construct (CSC), trajectories are solved using direct orthogonal collocation methods while leveraging navigation mesh techniques developed for fast geometric path planning solutions. To illustrate two-dimensional flight trajectories, sample results are applied to flight through downtown Chicago at an altitude of 600 feet above ground level. The three-dimensional problem is examined for feasibility by applying the methodology to a small scale problem. Computation and objective times are reported to illustrate the design implications for real-time optimal control systems, with results showing 86% reduction in computation time over traditional methods.

15. SUBJECT TERMS

Unmanned Vehicles, SUAS, Manned Unmanned Teaming, Path Planning, Optimal Control, Optimization,

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. R. G. Cobb, AFIT/ENY
U	U	U	U	223	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4559; richard.cobb@afit.edu