



**OPTIMAL UAV PATH PLANNING WITH
DYNAMIC NO-FLY-ZONES FOR TARGET
GEOLOCATION USING LINE-OF-BEARING
MEASUREMENTS AND KALMAN
FILTERING**

THESIS

Peter T. Heidlauf, Second Lieutenant, USAF
AFIT-ENY-MS-17-M-262

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENY-MS-17-M-262

OPTIMAL UAV PATH PLANNING WITH DYNAMIC NO-FLY-ZONES FOR
TARGET GEOLOCATION USING LINE-OF-BEARING MEASUREMENTS
AND KALMAN FILTERING

THESIS

Presented to the Faculty
Department of Aeronautical Engineering
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 Master of Science in Aeronautical Engineering

Peter T. Heidlauf, B.S.M.E.

Second Lieutenant, USAF

February 22, 2017

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENY-MS-17-M-262

OPTIMAL UAV PATH PLANNING WITH DYNAMIC NO-FLY-ZONES FOR
TARGET GEOLOCATION USING LINE-OF-BEARING MEASUREMENTS
AND KALMAN FILTERING

Peter T. Heidlauf, B.S.M.E.
Second Lieutenant, USAF

Committee Membership:

Dr. Richard Cobb
Chair

Dr. David Jacques
Member

Maj Garrison Lindholm, PhD
Member

Abstract

The Air Force Research Laboratory is developing an Unmanned Aerial Vehicle (UAV) to carry out the mission of Tactical Off-Board Sensing (TOBS). An optimal trajectory generation algorithm is needed for the semi-autonomous UAV to maximize the utility during its limited endurance while generating actionable target coordinates.

A method of dynamic UAV trajectory optimization was developed to geolocate a stationary target using Line-of-Bearing measurements and to approach an ideal orbit above the target in minimum time. Both dynamic and static No-Fly-Zones (NFZs) were enforced, and different wind conditions, initial conditions, and flight altitudes were simulated. A bootstrapping method was used which approached the true minimum final time solution by iteratively minimizing the target covariance at a specified time step and extending or trimming the state vector based on constraints in the top-level algorithm. A Kalman filter was implemented to estimate the target position to a 6 meter Circular Error Probable 90% covariance requirement. Using the characteristics of a candidate TOBS airframe, various real-world scenarios were modeled, optimized, and simulated to determine expected performance, optimal trajectory characteristics, and to provide a basis for the development of Tactics, Techniques, and Procedures for the TOBS program.

Through trajectory optimization and independent variation of scenario conditions, targeting times were decreased by up to 19% dependent on wind conditions, and up to 10% depending on launch positioning relative to NFZs. The optimal altitude was determined to be 1250 feet AGL for the simulated airframe, sensors, and slant range. Sustained target custody and covariance requirements under multiple continuous surveillance trajectories were simulated and verified.

Acknowledgments

I would like to thank my thesis adviser Dr. Richard Cobb for his instruction and oversight throughout the research process. His experience and recommendations were invaluable to my research. Thank you to my AFRL advisers, Major Garrison Lindholm, Dr. David Grymin, and Mr. Paul Fleitz, for their mentoring, encouragement, and technical aid throughout duration of my research.

Finally, I would like to thank my family, my fiancée Emily Barker, and my best friend Jules for supporting and encouraging me to persevere in my research through to the end.

Peter T. Heidlauf

Table of Contents

	Page
Abstract	iv
Acknowledgments	v
List of Figures	ix
List of Tables	xii
List of Acronyms	xiii
List of Symbols	xiv
I. Introduction	1
1.1 Motivation	1
1.2 Problem Statement	2
1.3 Assumptions	3
1.3.1 UAV Model	3
1.3.2 Initial Conditions and Known Information	4
1.3.3 Sensors and Measurements	5
1.3.4 Target and Terrain	5
1.4 Methodology	5
1.5 Overview	6
II. Background	8
2.1 Relevant Technologies	8
2.2 Line-of-Bearing Geolocation	9
2.2.1 Vision-Based Target Geolocation Using Micro Air Vehicles	9
2.2.2 Geolocation Using Direction Finding Angles	12
2.3 Estimation Methods	15
2.3.1 Full Batch Estimation	15
2.3.2 Kalman Filter	17
2.3.3 Comparison of Methods	19
2.4 Path Optimization	20
2.5 Constraining Final Covariance	21
2.6 Summary	22
III. Research Methodology	24
3.1 Chapter Overview	24
3.2 Line of Bearing Measurements	24
3.3 Converting Sensor Covariance to the Cartesian Frame	28

	Page
3.4 Targeting Kalman Filter	29
3.5 Varying Covariance for the Kalman Filter	31
3.6 Choosing a Constant Altitude for Optimization	34
3.7 Direct Method of Path Optimization	35
3.7.1 Cost Function	36
3.7.2 State Vector Formulation	39
3.7.3 Constraints	40
3.7.4 Gradients and Hessians	41
3.8 Bootstrapped Direct Method of Path Optimization	41
3.9 Other Calculations	43
3.10 Optimization Problem Setup	46
3.10.1 Continuous Setup	46
3.10.2 Discrete Setup	48
3.11 Simulations	49
3.12 Summary	49
IV. Results	50
4.1 Chapter Overview	50
4.2 Simulation Settings	50
4.3 Simplified Examples	54
4.4 Variation of the Cost Function	58
4.5 Optimal Solution to a Real-World Scenario	59
4.6 Comparison of Different Altitudes	64
4.7 Dependence on Initial Conditions	67
4.8 Effects of Constant Wind	69
4.9 Steady-State Target Observation	72
4.9.1 Steady-State Observation with an Orbit Obstruction	74
4.10 Algorithm Performance	77
4.11 Summary	80
V. Conclusions and Recommendations	82
5.1 Conclusions from Simulated Results	83
5.2 Identified Traits of Optimal UAV Trajectories	84
5.3 Conclusions on the Optimization Algorithm	85
5.4 Recommendations for Future Work	87
5.5 Closing Remarks	88
Appendix A. Cost Function MATLAB Code	90
Appendix B. No-Fly-Zone Constraints MATLAB Code	91

	Page
Appendix C. Line-of-Bearing Measurements and Approximation MATLAB Code	93
Appendix D. Discrete Kalman Filter MATLAB Code	97
Appendix E. Top-level Bootstrapped MATLAB Code	98
Appendix F. Point-Mass Model MATLAB Code	106
Bibliography	107

List of Figures

Figure		Page
1.1	A TOBS geolocation scenario with static and dynamic NFZs and a specified orbit	3
1.2	The point mass model displayed at sample times for illustration: t_1 , t_2 , and t_n	4
2.1	Small UAVs commonly used for surveillance and reconnaissance	8
2.2	Final location estimate and 95% error ellipse with dashed lines representing 2D Line-of-Bearing (LOB) measurements (Figure from [1])	15
2.3	Conceptual approach and flare segments (Figure from [2])	21
2.4	Flight paths with different levels of required final covariance (Figure from [2])	22
3.1	The sensor and ground target expressed in ENU coordinates (Raven graphic courtesy of AeroVironment Inc. [3])	25
3.2	The geometry of the LOB range calculation	27
3.3	Geometry of the conversion from angular units to distance units for LOB sensor covariances	29
3.4	The intersections of a plane (perpendicular to a sensor frame covariance axis) with the East-North-Up (ENU) axes are candidate covariance component approximations	32
3.5	The sensor frame covariances approximated as an ENU-aligned ellipsoid	33
3.6	The cost function weights control effort, target covariance, and distance from the ideal orbit	38
3.7	The bootstrapping algorithm as a system flow chart	42
3.8	The interactions of <code>Opt_Bootstrapper</code> , <code>fmincon</code> , and the other sub-functions	43

Figure		Page
3.9	The geometry of the dynamic No-Fly-Zone (NFZ) due to a target-orbiting aircraft	45
4.1	A clockwise orbit of the target at 500 ft AGL and slant range of 2000 ft	55
4.2	Four west-east flybys simulated at different offsets from the target	57
4.3	Comparison of flight paths for the cost functions outlined in Table 4.3	59
4.4	The geometry of a real-world scenario	60
4.5	The NFZs and trajectory of the Unmanned Aerial Vehicle (UAV) for a real-world scenario with wind.....	61
4.6	The optimal control vector, $\dot{\psi}$, for a real-world scenario with wind	62
4.7	The distance to each NFZ with respect to time for a real-world scenario with wind	62
4.8	The sensor and target covariances with respect to time for a real-world scenario with wind	63
4.9	Comparison of optimal flight paths at for several altitudes (detection and dynamic NFZs not shown).....	64
4.10	The sensor and target covariances with respect to time for a 1,250 ft AGL trajectory	66
4.11	The sensor and target covariances with respect to time for a 500 ft AGL trajectory	67
4.12	Comparison of the NFZs and trajectories for the initial conditions listed in Table 4.5	68
4.13	Comparison of the NFZs and trajectories for different wind conditions	69
4.14	Comparison of the control vectors for different wind conditions	70
4.15	Orbit of the target at 1,000 ft AGL in counter-rotation with a dynamic NFZ	72

Figure		Page
4.16	Orbit of the target at 1,000 ft AGL in co-rotation with a dynamic NFZ	73
4.17	Attempted orbit of the target at 500 ft AGL with an obstruction NFZ	74
4.18	The optimal control for the obstructed orbit	75
4.19	Distances to the NFZs with respect to time for the obstructed orbit	76
4.20	Sensor and target covariance for the obstructed orbit	77

List of Tables

Table		Page
2.1	Definitions of the Components of a Kalman Filter	17
2.2	Computational Efficiency of Estimators	19
4.1	Algorithm Variables, Values, and Descriptions	52
4.2	Specification of MATLAB <code>fmincon</code> Options	53
4.3	Variation of Cost Function Weights	58
4.4	The Effects of Different UAV Altitudes	65
4.5	Variation of Initial Conditions	68
4.6	Changes in Optimal Path and Targeting Time Due to Wind Conditions	71
4.7	Performance of the Bootstrapping Algorithm Under Different Settings	78

List of Acronyms

AFRL	Air Force Research Laboratory
CAS	Close Air Support
DCM	Direction Cosine Matrix
DTED	Digital Terrain Elevation Data
ECEF	Earth-Centered-Earth-Fixed
EKF	Extended Kalman Filter
ENU	East-North-Up
GPS	Global Positioning System
IMU	Inertial Measurement Unit
KF	Kalman filter
LIDAR	Light Detection And Ranging
LOB	Line-of-Bearing
NFZ	No-Fly-Zone
NLP	Nonlinear Programming
SLAM	Simultaneous Localization and Mapping
TOBS	Tactical Off-Board Sensing
TTPs	Tactics, Techniques, and Procedures
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
USAF	United States Air Force

List of Symbols

Notation used in Chapters III and IV		J	Cost function
		K	KF gain
$\bar{x}_j$	Position vector of subscript j	k	Discretized vector index
$\dot{\psi}$	Heading angle rate	N	Index of tf
$\dot{\psi}_{max}$	Maximum turn rate	N_{t1}	Index of $t1$
$\hat{x}$	Estimate of state	P	KF state covariance
$\hat{x}_t$	Estimate of target x-coordinate	p	Position
$\hat{y}_t$	Estimate of target y-coordinate	Q	KF system covariance
$\hat{z}_t$	Estimate of target z-coordinate	R	KF measurement covariance
ϕ	LOB Azimuth angle	R	Radius
ψ	Heading angle	r	Range
ψ_w	Wind angle	$t1$	Time to achieve $CovReq$
σ_ϕ	Covariance of ϕ measurement	tf	Final time
σ_θ	Covariance of θ measurement	u	KF input
σ_E	Covariance of East (x) axis	V	UAV airspeed
σ_N	Covariance of North (y) axis	V_w	Wind speed
σ_r	Covariance along r axis	w	Weight
σ_U	Covariance of Up (z) axis	x	Position x-coordinate
$\sigma_{x'}$	Covariance along x' axis	x'	Cartesian sensor frame azimuth axis
$\sigma_{z'}$	Covariance along z' axis	y	Position y-coordinate
θ	LOB Elevation angle	y'	Cartesian sensor frame range axis
$\tilde{z}$	KF Error between measurement and estimate	z	KF measurement
dt	Time step	z	Position z-coordinate
F	KF state matrix	z'	Cartesian sensor frame elevation axis
G	KF input matrix	$Z_{\frac{\alpha}{2}}$	Z-score for confidence interval
g	Inequality constraint	Subscripts	
H	KF output matrix	k	at current time step
h	Altitude	$k + 1 \mid k$	at next time step given current time step
I	Identity Matrix		

$k + 1$ at next time step

LOB Line-of-Bearing

NFZ No-Fly-Zone

t Target

v Vehicle

Superscripts

b Body frame

n ENU inertial frame

OPTIMAL UAV PATH PLANNING WITH DYNAMIC NO-FLY-ZONES FOR TARGET GEOLOCATION USING LINE-OF-BEARING MEASUREMENTS AND KALMAN FILTERING

I. Introduction

1.1 Motivation

Target geolocation has long been a requirement for military surveillance, law enforcement applications, and civilian conservation efforts. In situations where physical danger, geography, or time-sensitivity prevent ground-based target geolocation, a rapidly deployable Unmanned Aerial Vehicle (UAV) offers a swift targeting solution using on-board sensors and vision-based systems. Due to modern technology improvements, the potential for air-launched vehicles is a new alternative to traditional hand or ground-launched UAVs currently in use. These rapidly deployable UAVs offer a momentous aid to military operations by offering instantaneous capabilities of surveillance and target geolocation.

One example application of rapidly deployable UAVs is the United States Air Force (USAF) Close Air Support (CAS) mission. The targeting solutions necessary for effective air strikes require accurate geolocation and identification of the ground targets. If inclement weather or dangerous airspace prevents the geolocation and identification of a target, CAS assets are rendered ineffective. These situations require a new solution to conduct targeting and effectively carry out the CAS mission.

The Air Force Research Laboratory (AFRL) is developing a new technology to support the CAS mission named Tactical Off-Board Sensing (TOBS). The TOBS

system is an air-launched UAV designed to act as an off-board sensor. When adverse weather or standoff distance prevents a strike asset from having a direct line of sight to a ground target, the UAV will provide surveillance and targeting data from lower altitudes or closer ranges without endangering the higher-value asset. Due to the close proximity to ground targets and secondary air assets, the UAV must avoid any No-Fly-Zone (NFZ) present during operation. Because the TOBS system is designed to be operated as a sensor, the vehicle will be semi-autonomously flown with mission goals and target selection specified by an operator. New Tactics, Techniques, and Procedures (TTPs) must be developed for the UAV specific to the TOBS mission. Thus, the development of an autonomous trajectory-planning algorithm is a key step in maturing the TOBS system.

1.2 Problem Statement

Path planning algorithms are needed to efficiently guide the TOBS UAV as it carries out its mission. These path planning algorithms must take several unique factors into account. Due to the air-launched nature of the TOBS system, maximizing the utility of the limited flight endurance is a key priority. Additionally, the UAV must perform cooperatively with other air assets, staying clear of a secondary aircraft's weapon-target (or sensor-target) line at all times. Geolocation time, the time required to measure target coordinates with a specified level of statistical accuracy, can be minimized by optimally steering the UAV. Whether for military or civilian uses, an optimal path planning algorithm to minimize geolocation time in the presence of static and dynamic NFZs has many applications. A potential scenario is set forth in Figure 1.1.

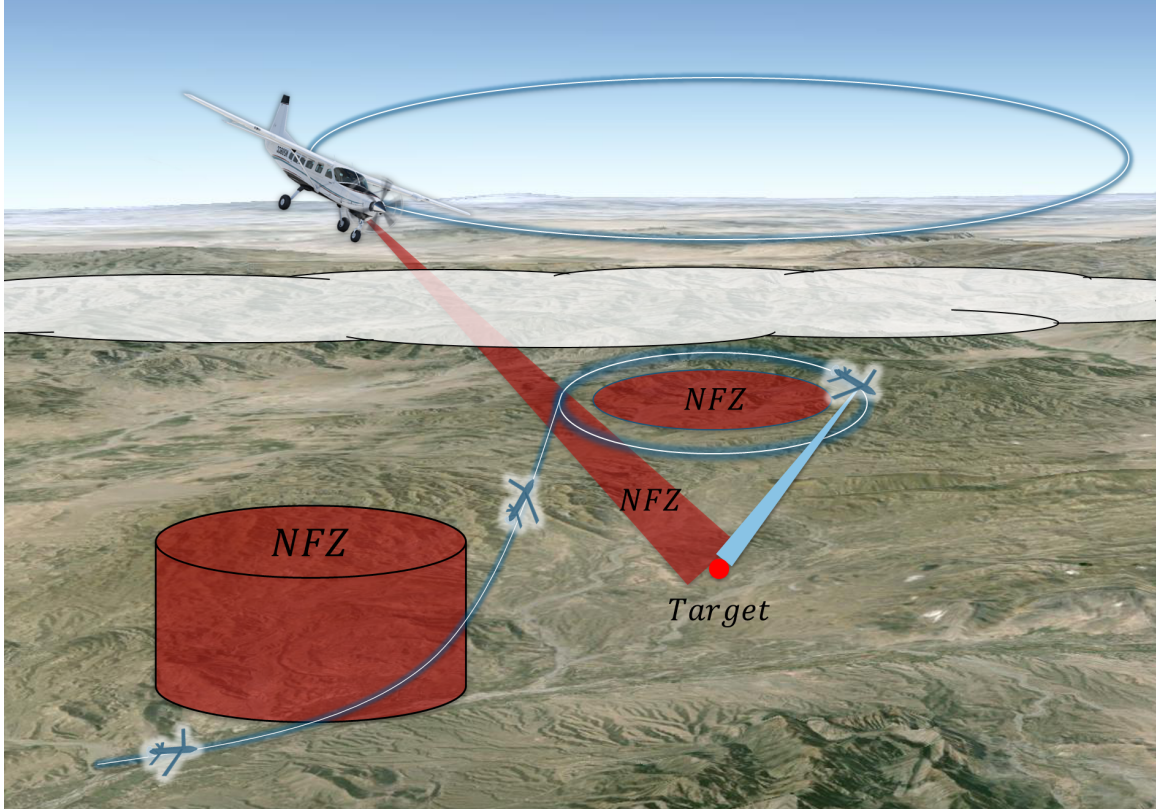


Figure 1.1. A TOBS geolocation scenario with static and dynamic NFZs and a specified orbit

The overall goal is to achieve a specified level of targeting accuracy using Line-of-Bearing (LOB) measurements and enter an appropriate orbit in minimum time given a dynamic no-fly zone and other relevant constraints.

1.3 Assumptions

1.3.1 UAV Model

In order to efficiently calculate optimal trajectories for the UAV, some simplifications and assumptions were made. Because the relevant airframes are slow and have limited range, the flat-Earth assumption was made to simplify the global coordinate system. An Earth-fixed East-North-Up (ENU) coordinate system was utilized to define the locations of all features of interest. Because the UAV was intended to

maximize endurance, extreme maneuvers (maneuvers with steep bank angles, high angle of attack, or high sideslip angle) were not included in the study. Thus, a constant speed, constant altitude point-mass model with heading rate as the control was implemented. This model is illustrated in Figure 1.2.

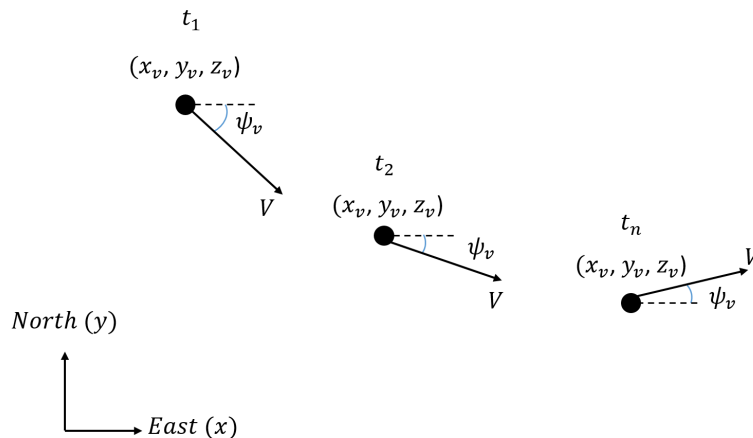


Figure 1.2. The point mass model displayed at sample times for illustration: t_1 , t_2 , and t_n .

The UAV position, heading, and heading rate are functions of time in this model, along with the dynamic (moving) NFZs. All other values (such as UAV altitude, z_v) were held constant.

1.3.2 Initial Conditions and Known Information

The TOBS operator inputs an initial target location at the beginning of a targeting sequence. The initial guess is assumed close enough to the actual target for the sensor to provide LOB measurements. The secondary aircraft's trajectory and all other sources of NFZs are assumed to be cooperative.

1.3.3 Sensors and Measurements

Due to the assumed on-board Global Positioning System (GPS) and Inertial Measurement Unit (IMU), the position and orientation of the UAV were assumed to be known at all times. The LOB sensor gimbal was assumed to have a full, unconstrained range of instantaneous motion due to the simplification of the sensor model. The LOB sensor directly coincided with the point-mass model position. No line of sight obstructions were included in the model, thus slant ranges and altitudes to prevent obstruction were selected. Due to the simulated nature of the problem, during the optimization algorithm the sensor always “measured” the LOB to the target perfectly. No noise in sensor measurements was included or simulated for algorithm convergence purposes. Real world errors in the sensor measurements were instead modeled as covariances in the azimuth and elevation angle measurements. These LOB angle measurements were the only source of uncertainty in this problem.

1.3.4 Target and Terrain

For this research, the target is assumed to be slow moving or static. The terrain is not required to be flat, but curvature of the Earth is neglected. The only constraint on terrain for this study is that it does not obstruct the UAV’s line of sight to the target or its flightpath. Although the methods developed could be used for a stationary or slow moving airborne target, the target is assumed to be on the ground in accordance with the TOBS mission.

1.4 Methodology

A method for converting LOB measurements to target coordinates with a quantifiable error ellipsoid was developed. A Kalman filter (KF) was designed to accept Cartesian position measurements of the target as inputs. The KF optimally calculated

and updated the measured target location and covariance matrix to fully define the target’s covariance ellipsoid (representing the uncertainty of the true target location).

An optimal path planning algorithm was developed using a direct method via a Nonlinear Programming (NLP) solver. The primary objective was that a specified level of targeting certainty be achieved in minimum time. Additionally, the algorithm was designed to be operable with or without LOB range measurements. Constraints including dynamic and static NFZs were included. Due to a fixed time step model, a bootstrapping method was created to approach a true minimum final time solution by iteratively minimizing target covariance and trimming or extending the state vector and final time.

Different variations of NFZs, wind conditions, altitudes, and initial conditions were simulated. Variations of the bootstrapping algorithm were studied to compare computational efficiency. The trajectories of a proposed airframe (with known sensor accuracies, air speeds, and turn limits) were simulated to predict real-world mission performance. These studies provided the baseline estimates for actual performance of the proposed UAV in the TOBS mission. Additionally, a qualitative study on the optimal flight trajectories determined trends in the flight path geometry. Given results of a variety of initial conditions and NFZ compositions, the optimal trajectory trends are presented for future use in developing a near-optimal trajectory generator for on-board usage and the development of TTPs for the TOBS program.

1.5 Overview

This chapter stated the objective of the thesis, established the motivation for research, and summarized the problem scope and key assumptions. Chapter II examines current airframes, the usage of LOB measurements, target estimation, the KF, and path planning in prior research. Chapter III develops and explains system modeling,

LOB measurement utilization, and bootstrapping algorithm methodology. Chapter IV describes the initial conditions and algorithm inputs selected for simulations, then analyzes the simulated results and discusses implications. Both quantitative and qualitative measurements are analyzed. Chapter V reviews the simulation findings, reiterates trends of the optimal trajectories to aid in the development of TOBS TTPs, and discusses recommendations for future work based on the research findings.

II. Background

2.1 Relevant Technologies

During operation, many surveillance-capable Unmanned Aerial Vehicle (UAV)s host the same basic suite of sensors: Global Positioning System (GPS), Inertial Measurement Unit (IMU), airspeed sensor, and a body-fixed or gimbal-mounted camera. For example, the Nighthawk Unmanned Aerial System (UAS) is a backpack portable system proven for its utility in search and rescue operations [4] and military reconnaissance. The Nighthawk is a 1.6 lb vehicle with a 26 inch wingspan, 40 mph cruise speed, and 11,000 foot flight ceiling [5]. When flown above 300 feet AGL, the Nighthawk, shown in Figure 2.1a, remains audibly undetectable. Similar systems such as the Pioria Maveric shown in Figure 2.1b and AeroVironment Raven B shown in Figure 2.1c also make use of electro-optical or IR cameras for surveillance capabilities [6]. The Maveric, Raven, and Nighthawk share similar performance specifications as they are small UAVs. They are easily hand-launched and fly autonomously, but have short flight times ranging from 20 to 60 minutes [5].



(a) ARA Nighthawk [7]



(b) Pioria Maveric [8]



(c) AeroVironment Raven [3]

Figure 2.1. Small UAVs commonly used for surveillance and reconnaissance

Larger and more advanced UAVs may host additional sensors including range finders, radar, Light Detection And Ranging (LIDAR), or other avionics systems. For the focus of this research, however, all sensor readings will be reduced to the form of periodic Line-of-Bearing (LOB) measurements (with and without range measurements) from the UAV to the ground target. This simplification allows for a wide application of the research to different airframes, sensor packages, and processing capabilities.

2.2 Line-of-Bearing Geolocation

In recent years, a plethora of research in robotics, controls, and automation has explored the capabilities of LOB measurements in mapping, localization, and target geolocation. Several relevant articles were studied to gain context for the research and to review prior methods and solutions to the problems involved in geolocation.

2.2.1 Vision-Based Target Geolocation Using Micro Air Vehicles

In 2008, Pachter et al. developed and analyzed a method of locating a stationary ground target using vision-based bearing measurements from micro air vehicles [9]. Bearing-only measurements of a ground object were obtained using an image processing algorithm to track a manually selected feature. Let n represent the navigation frame, b the body frame, and P the ground target, Equations 2.1 and 2.2 define the position of the ground target relative to the origin of the navigation frame.

$$R_P = R_v + R_{LOB} \quad (2.1)$$

$$R_P^n = R^n + C_b^n R_{LOB}^b \quad (2.2)$$

The locations of the target and vehicle are R_P and R_v , respectively. The LOB vector is R_{LOB} . Equation 2.3 defines the Direction Cosine Matrix (DCM) C_b^n given the vehicle's Euler angles, where ϕ , θ , and ψ represent bank, pitch, and heading angles. The DCM transforms a vector from the body frame b to the navigation frame n . Note that $(C_b^n)^{-1} = C_n^b$ and transforms from the navigation frame to the body frame.

$$C_b^n = \begin{bmatrix} \cos \psi \cos \theta & \cos \psi \sin \theta \sin \phi - \sin \psi \cos \phi & \sin \psi \sin \phi + \cos \psi \sin \theta \cos \phi \\ \sin \psi \cos \theta & \cos \psi \cos \phi + \sin \psi \sin \theta \sin \phi & \sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi \\ -\sin \theta & \cos \theta \sin \phi & \cos \theta \cos \phi \end{bmatrix} \quad (2.3)$$

Effectively, knowing the navigation state of the UAV $(x, y, z, \phi, \theta, \psi)$ generates R^n and C_b^n . Given a ranged LOB measurement from the vehicle to the ground target, R_{LOB} , the target location could be calculated immediately. This algorithm relied on a camera's focal plane and focal length, f_l , to calculate a range measurement [9]. This allowed for Equation 2.4 to calculate the coordinates of the ground target given the UAV state and measurement from the camera frame. The complete derivation was shown in [9].

$$\begin{bmatrix} x_P \\ y_P \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + (z_P - z) \frac{1}{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} C_b^n \begin{bmatrix} x_f & y_f & f_l \end{bmatrix}^T} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} C_b^n \begin{bmatrix} x_f \\ y_f \\ f_l \end{bmatrix} \quad (2.4)$$

The measurements x_f and y_f are the target coordinates in the camera frame. The target altitude z_P is assumed to be known. Methods for both linear and nonlinear regression were developed to estimate the problem parameters given multiple measurements of the target. The linear regression method for the estimation of measured parameters is reviewed below. The measured variables are $y_1 = [x, y, z, x_f, y_f]^T$ and $C_b^n = C_b^n(\phi, \theta, \psi)$. The parameters to be estimated are shown in Equations 2.5

through 2.7. Note that f in Equation 2.6 specified a nonlinear function.

$$\theta = [\theta_1, \theta_2]^T \quad (2.5)$$

where

$$\theta_1 = [x_P, y_P]^T = f(y_1, C_b^m) \quad (2.6)$$

and

$$\theta_2 = [\overline{\delta\alpha}, \overline{\delta\beta}, \overline{\delta\gamma}]^T \quad (2.7)$$

The bias errors in the attitude measurement from the UAV's sensors are $\overline{\delta\alpha}$, $\overline{\delta\beta}$, and $\overline{\delta\gamma}$ [9]. These are not biases in the Euler angles. Due to the linearizations made with this method, it was assumed that measurement errors were small and that systematic errors were constant. In Equation 2.5, θ_2 quantified a small-angle rotation misalignment from the attitude-heading reference system [9].

A_k and B_k (Equations 2.8 and 2.9) are the partial derivatives of f evaluated at time k using the measured variables z_1 and $C_{b_{c_k}}^n$ [9].

$$A_k \equiv \left. \frac{\partial f}{\partial y_1} \right|_{z_1, C_{b_{c_k}}^n} \quad (2.8)$$

$$B_k \equiv \left. \frac{\partial f}{\partial \theta_2} \right|_{z_1, C_{b_{c_k}}^n} \quad (2.9)$$

The closed-form solution of the linear regression is expressed in Equation 2.10.

$$\hat{\theta} = \left[\sum_{k=1}^N \begin{bmatrix} I_2 \\ B_k^T \end{bmatrix} (A_k R_1 A_k^T + \sigma^2 B_k B_k^T)^{-1} [I_2, B_k] \right]^{-1} \times \sum_{k=1}^N \begin{bmatrix} I_2 \\ B_k^T \end{bmatrix} (A_k R_1 A_k^T + \sigma^2 B_k B_k^T)^{-1} f(z_{1_k}, C_{b_{c_k}}^n) \quad (2.10)$$

Let R_1 be a 5×5 real symmetric positive-definite matrix: the Gaussian measurement noise covariance. The calculation for the parameter's estimation error covariance is shown in Equation 2.11.

$$P = \left[\sum_{k=1}^N \begin{bmatrix} I_2 \\ B_k^T \end{bmatrix} (A_k R_1 A_k^T + \sigma^2 B_k B_k^T)^{-1} [I_2, B_k] \right]^{-1} \quad (2.11)$$

This method of linear regression yielded a non-iterative solution for the parameter estimates of x_p , y_p , and the small angle rotation misalignments as well as the error covariance of the estimates. The author stated “because the parameter estimate is obtained via the solution of a linear regression, the parameter estimate is not biased; that is, the target is accurately geolocated. Moreover, a reliable data-driven estimate P of the localization error is also obtained” [9].

2.2.2 Geolocation Using Direction Finding Angles

Grabbe and Hamschin developed an algorithm that utilized iterated least-squares to estimate a stationary ground target's position using multiple angle measurements [1]. This research used an Earth-Centered-Earth-Fixed (ECEF) coordinate frame for all angle measurements and utilized Digital Terrain Elevation Data (DTED) to determine the altitude of the target. The target position (in ECEF coordinates) was

defined in Equation 2.12. Note that r_{east} was the Earth's transverse radius of curvature, ϵ was Earth's eccentricity, and r_{eq} was Earth's equatorial radius, and ψ and θ were the WGS84 longitude and geodetic latitude to be found [1]. The relationship between range, direction finding angles, and the target position is expressed in Equation 2.12.

$$P_{tgt} = \begin{bmatrix} (r_{east} + a) \cos(\psi) \cos(\theta) \\ (r_{east} + a) \sin(\psi) \cos(\theta) \\ (r_{east}(1 - \epsilon^2) + a) \sin(\theta) \end{bmatrix} \quad (2.12)$$

The unit vector, A_u , lies along the line of sight to the target. It is calculated in Equation 2.13. The “direction finding” angles, ϕ and η (azimuth and elevation), are defined in Equations 2.14 and 2.15.

$$A_u = \begin{bmatrix} \cos(\phi) \cos(\eta) \\ \sin(\phi) \cos(\eta) \\ -\sin(\eta) \end{bmatrix} \quad (2.13)$$

$$\phi = \tan^{-1} \left(\frac{A_u(2)}{A_u(1)} \right) \quad (2.14)$$

$$\eta = \tan^{-1} \left(\frac{-A_u(3)}{\sqrt{A_u(1)^2 + A_u(2)^2}} \right) \quad (2.15)$$

The estimated state is defined in Equation 2.16.

$$x = \begin{bmatrix} \psi \\ \theta \end{bmatrix} \quad (2.16)$$

In Equation 2.17, v_i is assumed to be the unbiased Gaussian error in a direction finding angle measurement.

$$z_i = h_i(x) + v_i \quad (2.17)$$

R is the $n \times n$ positive definite measurement error covariance matrix where n is the number of direction finding measurements taken [1]. Each σ_i is the angle error of a measurement.

$$R = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sigma_n^2 \end{bmatrix} \quad (2.18)$$

It can be shown that Equation 2.19 evaluates the covariance matrix $P_{p_{tgt}}$ of the estimated target position p_{tgt} . The specific components of Equation 2.19 were described fully by Grabbe and Hamschin [1].

$$P_{p_{tgt}} = \left[\frac{\partial p_{tgt}}{\partial x} \right] P_x \left[\frac{\partial p_{tgt}}{\partial x} \right]^T \quad (2.19)$$

The benefit of calculating the $P_{p_{tgt}}$ matrix was explained by Grabbe and Hamschin as:

To construct an error ellipse, the above ECEF covariance matrix is rotated into the coordinate frame used by the signals intelligence operators workstation, the appropriate 2×2 submatrix is extracted, and an eigen decomposition is performed to determine the length and direction of the ellipse axes. [1]

Once the target location was estimated and the covariance $P_{p_{tgt}}$ was calculated, the “error ellipse” was found using an eigen decomposition. Sample results of this calculation from the article are shown in Figure 2.2.

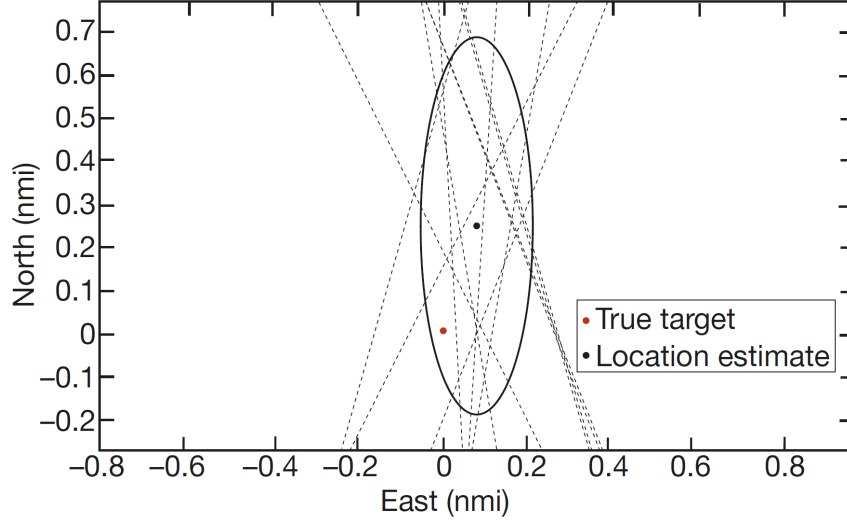


Figure 2.2. Final location estimate and 95% error ellipse with dashed lines representing 2D LOB measurements (Figure from [1])

2.3 Estimation Methods

Two primary methods for estimation of the target's position and covariance were considered for this research: full batch and recursive estimation. Both methods require measurements of the state and a knowledge of the sensor covariance to operate, as well as an initial state and error covariance for the estimator itself. One method of linear batch regression of measured parameters was presented in Section 2.2.1.

2.3.1 Full Batch Estimation

Batch estimation was a calculation done after the collection of a full set of measurements. Linear least-squares regression and extended least-squares regression were both common examples of batch estimation. The benefit of batch estimation is its consideration of every measurement in the history of the problem in calculating the state at each step [10]. The disadvantages of batch estimation are the costs of real-time operation and the required memory. For tracking in real time, the batch algorithm was re-processed after each new measurement. For a small number of measurements,

this had negligible effect on efficiency. However, as the number of measurements increased, the processing time required increased significantly.

The method of batch estimation by linear least-squares regression minimized the sum of squared differences between the measurements and the determined state. A simple example of this method was shown by solving the over-determined system Equation 2.20 as shown in Equation 2.21 [11]. Here, $\hat{x}$ (an $n \times 1$ vector) was the estimated state given the set of measurements, y (an $m \times n$ vector), and the state dynamics, A (an $n \times n$ matrix). Note each row of y corresponds to a single measurement of the m states.

$$Ax = y \tag{2.20}$$

$$\hat{x} = (AA^T)^{-1}A^Ty \tag{2.21}$$

Computing the solution to the normal equation (solving the least-squares problem) had a quantifiable processing cost. Let m be the number of measurements and n be the number of system states. As defined in [11], a “flop” is one matrix multiplication operation and one matrix addition operation. If the normal equations method was used to solve the least-squares problem, the cost of calculation (using the common singular value decomposition method) for large m and n is $mn^2 + \frac{1}{3}n^3$ flops [11]. However, if this calculation was completed at every time step or measurement, the total cost rises to $m^2n^2 + \frac{1}{3}mn^3$ flops.

If $n \ll m$, the iteration and total costs change. The iteration cost was approximated as m flops, and the total cost as m^2 flops. In batch least-squares regression, n must be less than or equal to m to determine accurate results. Until the m th measurement, the system was under-determined.

2.3.2 Kalman Filter

A recursive estimator stores no memory of previous measurements. Instead, the estimate of the current state and its covariance are stored and updated by the system dynamics and measurements. The Kalman filter (KF) is one form of a recursive estimator, and is optimal in the least-squares sense for linear Gaussian processes [10]. For nonlinear problems, the standard KF may be insufficient, but could be replaced in the form of an extended or unscented KF as shown by Ross in his 2D trajectory optimization based on LOB measurements [12] and by Deans in the development of Simultaneous Localization and Mapping (SLAM) algorithms [10].

The equations for a discrete KF are shown symbolically below in Equation [13]. The symbolic definitions and notation meanings were displayed in Table 2.1.

Table 2.1. Definitions of the Components of a Kalman Filter

Variable	Definition
$\hat{x}$	Estimate of state
P	State covariance
F	State transition matrix
G	Input matrix
H	Output matrix
Q	System covariance
R	Measurement covariance
u	Input
z	Measurement
K	Kalman gain
k	At current time step (subscript)
$k + 1$	At next time step (subscript)
$k + 1 k$	At next time step given current time step (subscript)

Equations 2.22 and 2.23 are the filter's prediction steps, where the state estimate and covariance at the next time step are calculated using the system's dynamics and

knowledge of the current time step.

$$\hat{x}_{k+1|k} = F_k \hat{x}_{k|k} + G_k u_k \quad (2.22)$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k \quad (2.23)$$

Equation 2.24 calculates the error between the measurement and state.

$$\tilde{z}_{k+1} = z_{k+1} - H_{k+1} \hat{x}_{k+1|k} \quad (2.24)$$

Equations 2.25 and 2.26 are the correction steps, where the estimate and covariance are updated using the Kalman gain and error from the new measurement.

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1} \tilde{z}_{k+1} \quad (2.25)$$

$$P_{k+1|k+1} = [I - K_{k+1} H_{k+1}] P_{k+1|k} [I - K_{k+1} H_{k+1}]^T + K_{k+1} R_{k+1} K_{k+1}^T \quad (2.26)$$

Equation 2.27 is the calculation of the Kalman gain.

$$K_{k+1} = P_{k+1|k} H_{k+1}^T [H_{k+1} P_{k+1|k} H_{k+1}^T + R_{k+1}]^{-1} \quad (2.27)$$

The computation of a single step of a KF also has a measurable cost. Let p be the number of states in the measurement vector. The full cost of a KF iteration where $p \neq n$ is $3n^3 + 3np^2 + \frac{3}{2}np^2 + \frac{1}{6}p^3$ flops [14]. If every state is measurable ($p = n$), then the cost is $7\frac{2}{3}n^3$ flops. Finally, if the KF is iterated for each of m , full-state measurements, the total cost is $7\frac{2}{3}mn^3$.

2.3.3 Comparison of Methods

Both batch and iterative methods of approximation have unique advantages and costs depending on the application. The approximate computational costs of each method reviewed are compiled in Table 2.2. Note that Big- Ω notation shows a lower bound on the cost of an algorithm.

Table 2.2. Computational Efficiency of Estimators

Method	Ratio	Iteration Cost (flops)	Total Cost (flops)	Big- Ω
Least-Squares	$n \approx m$	$mn^2 + \frac{1}{3}n^3$	$m^2n^2 + \frac{1}{3}mn^3$	$\Omega(m^4)$
Least-Squares	$n \ll m$	m	m^2	$\Omega(m^2)$
Kalman Filter	$n \approx m$	$7\frac{2}{3}n^3$	$7\frac{2}{3}mn^3$	$\Omega(m^4)$
Kalman Filter	$n \ll m$	$7\frac{2}{3}$	$7\frac{2}{3}m$	$\Omega(m)$

For applications where the number of states was similar in size to the number of measurements, both least-squares regression and Kalman filtering offer a similar computational cost. However, as the number of measurements increases, the computational cost of the function grows at the rate of Big- Ω or faster [15]. Thus, the iterative Kalman filter is more efficient than the batch-processing of the least-squares method for large numbers of measurements. As well, the Kalman filter requires less memory in the same situation.

Nonetheless, research comparing the performance (rather than computational efficiency) of Batch-Least-Squares and extended Kalman filtering for radar tracking discovered that the batch method converged more accurately and reduced error faster than the Extended Kalman Filter (EKF) [16]. Thus, the selection of a batch or iterative estimation method is reduced to a choice between accuracy and computational efficiency. In the application of a computationally difficult optimization algorithm, the improved efficiency of a Kalman filter appears to be the most crucial factor.

2.4 Path Optimization

In 2008, Jorris and Cobb developed a multiple method 2-D trajectory optimization [17]. The goal of minimizing the time to reach a target while obeying path and no-fly zone constraints was achieved using a dynamic optimization method. The general form of the cost function J (which maps the multidimensional state vector and control inputs to a single scalar value) is expressed in Equation 2.28.

$$J = \phi(\bar{x}(t_0), \bar{x}(t_f), t_0, t_f) + \int_{t_0}^{t_f} L(\bar{x}(t), u(t), t) dt \quad (2.28)$$

To minimize final time, the function is evaluated with $\phi = t_f$ and $L = 0$ in Equation 2.29.

$$J = t_f \quad (2.29)$$

The No-Fly-Zone (NFZ) constraints were enforced using inequality constraints as shown in Equation 2.30. Each NFZ, j , was a circular region of radius R_j centered around (x_{cj}, y_{cj}) . The position of the vehicle at time t was $(x_v(t), y_v(t))$.

$$S(x(t), t) = \frac{1}{2} \left(R_j^2 - (x_v(t) - x_{cj})^2 - (y_v(t) - y_{cj})^2 \right) \leq 0 \quad j = 1, 2, \dots \quad (2.30)$$

For the constraints to be satisfied, every inequality constraint $S(x(t), t) \leq 0$ at every time step. Equality constraints are used to enforce the intermediate and final waypoint requirements. The final position constraint is expressed in Equation 2.31. The desired final position is (x_f, y_f) . For the equality constraint to be satisfied, $\psi(\bar{x}(t_f), t_f) = 0$ [17].

$$\psi(\bar{x}(t_f), t_f) = \begin{bmatrix} x(t_f) - x_f \\ y(t_f) - y_f \end{bmatrix} \quad (2.31)$$

The cost function, equality constraints, and inequality constraints are the definitions necessary for the dynamic optimization problem to be solved using a Nonlinear Programming (NLP) solver.

2.5 Constraining Final Covariance

By varying the cost function and constraint equations used by an NLP solver, the framing of the problem changes. A different cost function could change the optimization from a min-final-time problem to a weighted minimization of both final time and final covariance. In 2014, Ross and Cobb presented a trajectory optimization to minimize the final time required to achieve a specified target covariance using bearing-only target analysis [12]. The application of this method was to “land on a wire” using a small UAV or quad-copter as shown in Figure 2.3. A 2-D flight trajectory was optimized to achieve the end goal of landing on a wire located with a required certainty in minimum time. The weighted cost function Ross developed is shown in Equation 2.32 [2].

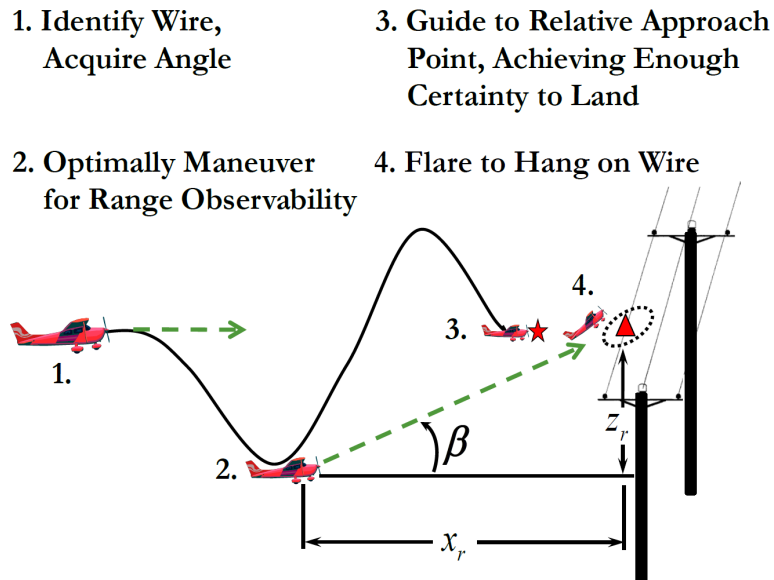


Figure 2.3. Conceptual approach and flare segments (Figure from [2])

$$J_{subopt} = w_t t_f + \int_{t_0}^{t_f} -w_x \sin^2 \beta - w_z \cos^2 \beta + u_y^T W_{u_y} u_y dt \quad (2.32)$$

The weight on controls, W_{u_y} , is held constant. The most relevant components of this cost function were the weights w_t , w_x and w_z . The weighting given to final time t_f was expressed as w_t . The weighting given to the final covariance of the target wire was w_x . Note that $w_z = 1 - w_x$. By varying the two weights w_t and w_x , different solutions were found by the NLP solver. Example solutions from the dissertation are shown in Figure 2.4. By relaxing the weight on final time from $w_t = 2$ to $w_t = 0.5$, a trajectory which achieved a higher level of certainty was found. This cost function did not guarantee that a specific level of certainty was reached, but still yielded valuable results. Instead, the final covariance requirement was guaranteed using an inequality constraint.

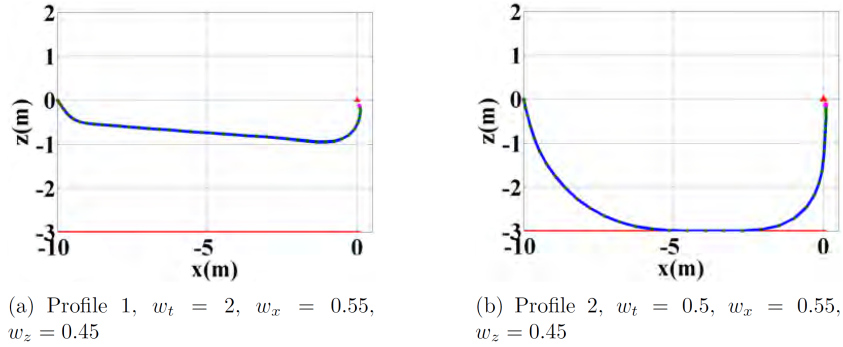


Figure 2.4. Flight paths with different levels of required final covariance (Figure from [2])

2.6 Summary

Taking into account prior research in the area of path optimization and LOB-only geolocation, several key methods and tools are available as a starting point. Given linear measurements of the target's position, a Kalman filter will iteratively estimate the target location and covariance. The selection and weighting of the cost

function and constraint equations will enforce the system dynamics and requirements to avoid NFZs. These tools and techniques are key in the development of the optimal trajectory-planner.

III. Research Methodology

3.1 Chapter Overview

Given a periodic source of Line-of-Bearing (LOB) measurements (with or without range measurements), a method of geolocating the target and evaluating the covariance for the measured target location was developed. A path optimization algorithm to minimize the amount of time (and the control effort and other potential factors) needed to achieve a required level of target accuracy and enter a specified orbit about the target. Furthermore, the optimization algorithm included dynamic and static NFZs, and accounted for wind in the point-mass model.

3.2 Line of Bearing Measurements

In either simulation or actual testing, a method of generating the actual LOB measurements was required. In simulation, perfect sensors were modeled using the “unknown” fixed target location p_t and the known location of the sensor to calculate a ranged LOB measurement. It was assumed that the sensor location coincided with the vehicle, p_v . A flat-Earth-fixed East-North-Up (ENU) coordinate system was used to define the locations of all features in simulation. Equation 3.1 is used to calculate the position of the target relative to the vehicle using a flat-Earth, fixed Cartesian coordinate frame (denoted by the superscript n).

$$p_{LOB}^n = p_t^n - p_v^n \quad (3.1)$$

The measurement represented in spherical coordinates in the body frame b is defined as the vector p_{LOB}^b . This measurement can be converted to spherical coordinates to represent the LOB measurement in terms of ϕ , θ , and r : the azimuth, elevation

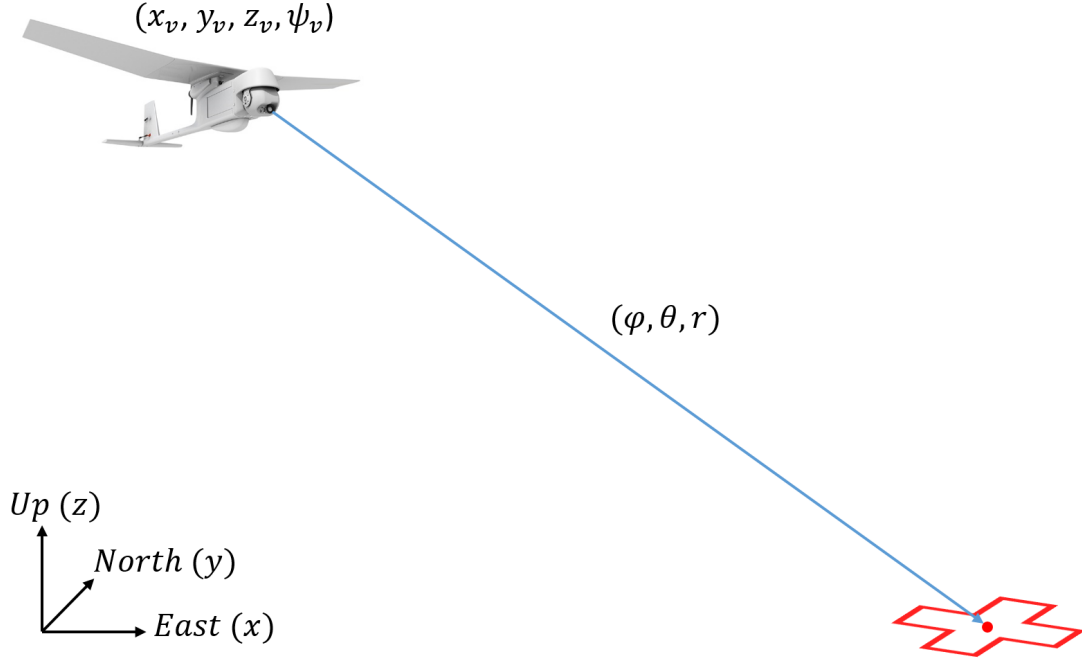


Figure 3.1. The sensor and ground target expressed in ENU coordinates (Raven graphic courtesy of AeroVironment Inc. [3])

angle, and range, respectively. The conversion is made in Equation 3.2 or with the MATLAB command `cart2sph`. The ENU components of p_{LOB}^b are x , y , and z .

$$\begin{bmatrix} \phi \\ \theta \\ r \end{bmatrix} = \begin{bmatrix} \tan^{-1} \left(\frac{y}{x} \right) \\ \tan^{-1} \left(\frac{z}{\sqrt{x^2 + y^2}} \right) \\ \sqrt{x^2 + y^2 + z^2} \end{bmatrix} \quad (3.2)$$

The conversion from spherical coordinates to Cartesian coordinates is calculated in Equation 3.3 or using the MATLAB command `sph2cart`.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r \cos \theta \cos \phi \\ r \cos \theta \sin \phi \\ r \sin \theta \end{bmatrix} \quad (3.3)$$

The body frame is aligned with the ENU coordinate system with its origin at p_v^n . If rangeless LOB measurements were modeled, r was set equal to 1. The rangeless LOB vector is $\hat{p}_{LOB}$. If sensor bias or sensor noise is simulated, the errors are included using Equation 3.4.

$$p_{LOB\ with\ error}^b = p_{LOB}^b + \begin{bmatrix} \phi_{error} \\ \theta_{error} \\ r_{error} \end{bmatrix} \quad (3.4)$$

The addition of sensor noise or bias can be used when simulating the measurements from a known flight path, however only a fixed sensor bias could be implemented during path optimization because of convergence issues. Essentially, as the glsnlp solver evaluates the cost function (and approximates gradients and Hessians) for many possible state vectors, noise that carries through to the cost function can prevent the solver from converging on the optimal solution.

If no range-measurement was available, a method of estimating the range from the sensor to the target was required to generate measurements of the ground target's position. Equation 3.5 is used to calculate the range along the line of bearing at which the perpendicular distance to the current estimate of the target position, $\bar{x}_t$, is minimized [18]. The Cartesian coordinates of three points are required for this calculation: the origin of the line of bearing, $\bar{x}_v$, an arbitrary point on the line of bearing, $\bar{x}_2 = \bar{x}_v + P_{LOB}$, and the estimate of the target position. The range from $\bar{x}_v$ along the LOB closest to the target estimate is defined as r . Once the range is measured or calculated, the complete measurement of the ground target's position in ENU can be calculated using Equation 3.6.

$$r = -\frac{(\bar{x}_v - \bar{x}_t) \cdot (\bar{x}_2 - \bar{x}_v)}{|\bar{x}_2 - \bar{x}_v|^2} \quad (3.5)$$

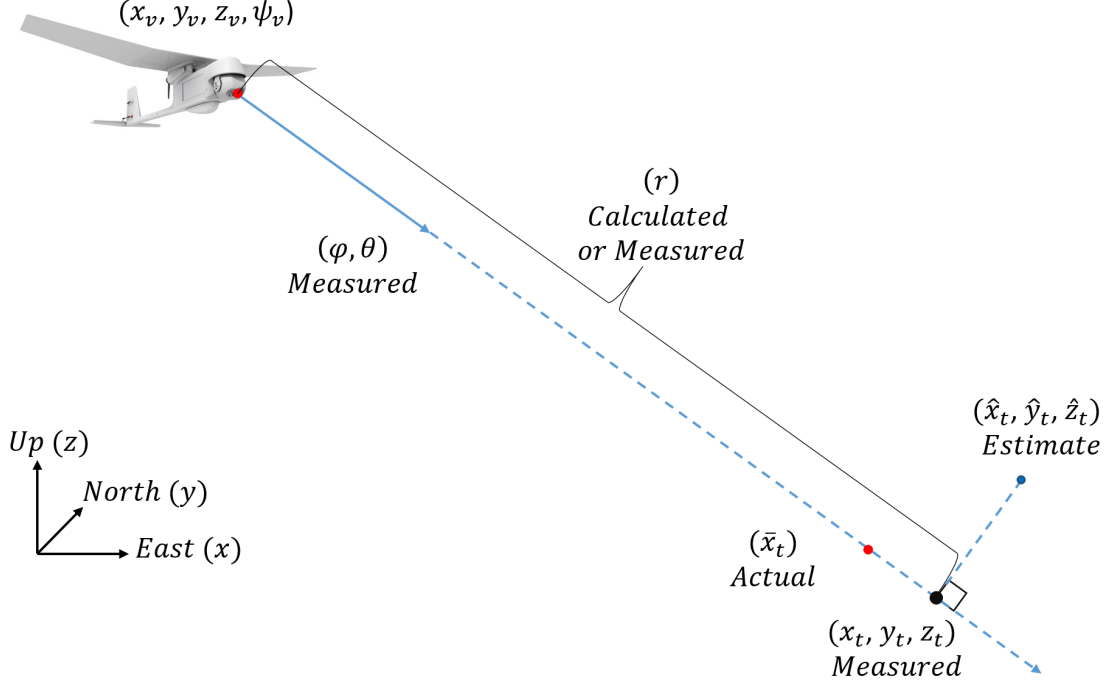


Figure 3.2. The geometry of the LOB range calculation

$$\bar{x}_{t,measured} = \bar{x}_v + \begin{bmatrix} r \cos \theta \cos \phi \\ r \cos \theta \sin \phi \\ r \sin \theta \end{bmatrix} \quad (3.6)$$

This method was implemented as using the functions `getLOB` and `LOB2xyzWithCov` included in Appendix C. With this implementation, if the target position estimate is perfect, the calculated range is also perfect. However, if the target position estimate is not perfect, the calculated range is close to the actual range but is not perfect. When combined with the Kalman filter (KF), the calculated ranges converge to the actual range after multiple measurements. With these functions developed, a method to convert the LOB sensor covariance to usable inputs is required.

3.3 Converting Sensor Covariance to the Cartesian Frame

Within the Unmanned Aerial Vehicle (UAV), one source of sensor covariance is considered. The LOB sensor is assumed to have zero-mean, Gaussian, and uncorrelated covariances in ϕ , θ , and r . The measurements and covariances of r are in distance units, and those of LOB ϕ and θ are in angular units.

A method for converting these covariances to a usable input to the targeting estimator is required. The geometric relationships between σ_ϕ , σ_θ , and r enabled a trigonometric conversion from angular covariance to a distance covariance represented in the body frame. The calculations are shown in Equations 3.7 and 3.8. Figure 3.3 visually represents the conversion. These equations assumed r was large enough such that σ_θ was unaffected by differences from Equation 3.5.

$$\sigma_{x'} = 2r \tan \frac{\sigma_\phi}{2} \quad (3.7)$$

$$\sigma_{z'} = 2r \tan \frac{\sigma_\theta}{2} \quad (3.8)$$

The value σ_r is only available if a range sensor is utilized. Otherwise, σ_r is assumed to be an arbitrarily large value. One significant issue that arose due to the LOB measurements was its initial incompatibility with a simple Kalman filter. Solutions to this compatibility issue are presented after the review of the Kalman filter's implementation. This method was also encompassed in the function `LOB2xyzWithCov`, included in Appendix C.

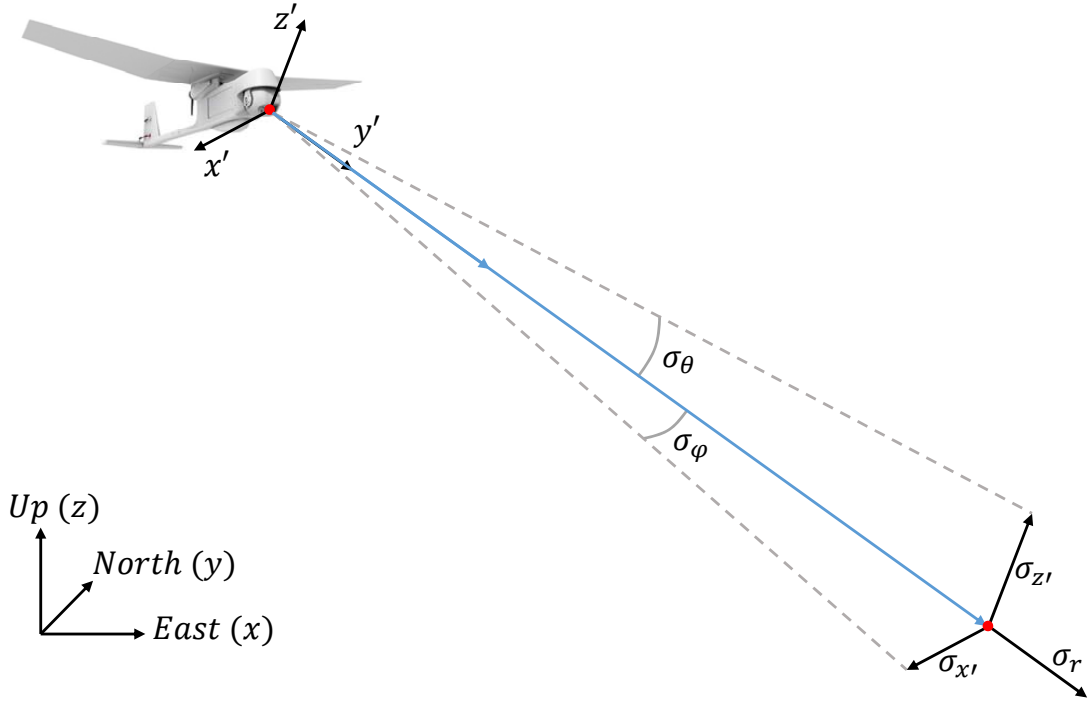


Figure 3.3. Geometry of the conversion from angular units to distance units for LOB sensor covariances

3.4 Targeting Kalman Filter

A discrete Kalman filter was implemented to iteratively estimate the position of the target in the ENU coordinate system. This method of estimation offered an optimal method (in the least squares sense) which could function both in simulation and real-time operation. The state to be estimated (for a small, stationary target) is described in Equation 3.9. Note that x_t , y_t , and z_t are the East, North, and Up coordinates of the target.

$$\hat{x} = \begin{bmatrix} x_t & y_t & z_t \end{bmatrix}^T \quad (3.9)$$

The dynamics of the system were represented in the matrix F_k as an identity matrix (I_3). The input matrix, G_k , was zero. Effectively this tracked the location of the target only through measurements of the target. Similarly, the measurement matrix

H_k is modeled as an identity matrix (I_3). Equations 3.10 through 3.15 are copied directly from Section 2.3.2 for the reader's reference.

$$\hat{x}_{k+1|k} = F_k \hat{x}_{k|k} + G_k u_k \quad (3.10)$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k \quad (3.11)$$

$$\tilde{z}_{k+1} = z_{k+1} - H_{k+1} \hat{x}_{k+1|k} \quad (3.12)$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1} \tilde{z}_{k+1} \quad (3.13)$$

$$P_{k+1|k+1} = [I - K_{k+1} H_{k+1}] P_{k+1|k} [I - K_{k+1} H_{k+1}]^T + K_{k+1} R_{k+1} K_{k+1}^T \quad (3.14)$$

$$K_{k+1} = P_{k+1|k} H_{k+1}^T [H_{k+1} P_{k+1|k} H_{k+1}^T + R_{k+1}]^{-1} \quad (3.15)$$

Equations 3.10 through 3.15 are implemented in the function **StepDKF** which receives Q_k , R_{k+1} , $P_{k|k}$, $\hat{x}_{k|k}$, and z_{k+1} as inputs. **StepDKF** is attached in Appendix D. The function evaluates and outputs the estimated state and system covariance, $\hat{x}_{k+1|k+1}$ and $P_{k+1|k+1}$ respectively. This iterative implementation allows for a changing sensor covariance matrix R_{k+1} . As well, this implementation is computationally efficient, which reduces the run time of the overall algorithm as shown in Subsection 2.3.3.

3.5 Varying Covariance for the Kalman Filter

The LOB measurements are converted to ENU target coordinates with sensor covariances represented in the sensor frame (x', r, z') . The discrete Kalman filter receives ENU coordinates as measurements and the sensor covariance matrix, R_{k+1} , expressed in the ENU frame. The conversion of covariances from the sensor frame to their equivalent expression in R_{k+1} is necessary to solve the compatibility issue mentioned in Section 3.3. The general format for R_{k+1} is expressed in Equation 3.16.

$$R_{k+1} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix} \quad (3.16)$$

In three dimensions, both R_{k+1} and $P_{k+1|k+1}$ can be represented by ellipsoids. The $P_{k+1|k+1}$ ellipsoid's principal axes are aligned with the ENU coordinate system.

The method developed for converting the sensor covariance from the sensor frame to the ENU frame is a linearizing approximation based on the geometry of the LOB measurements. Essentially, a plane perpendicular to each of the covariance components in the sensor frame intersects with the ENU axes. The intersections of the three planes (one each for x' , y' , and r) with the ENU axes represents the equivalent covariance components for R_{k+1} . This is illustrated in Figure 3.4. With exceptions for LOB measurements taken from due north-south, east-west, or directly above, there were 3 candidate covariances for East and North, and 2 candidate covariances for Up. Equations 3.17 through 3.20 detail the calculation of these points.

$$\begin{bmatrix} \sigma_{E1} \\ \sigma_{N1} \end{bmatrix} = \begin{bmatrix} \frac{\sigma_{x'}}{\sin \phi} \\ \frac{\sigma_{x'}}{\cos \phi} \end{bmatrix} \quad (3.17)$$

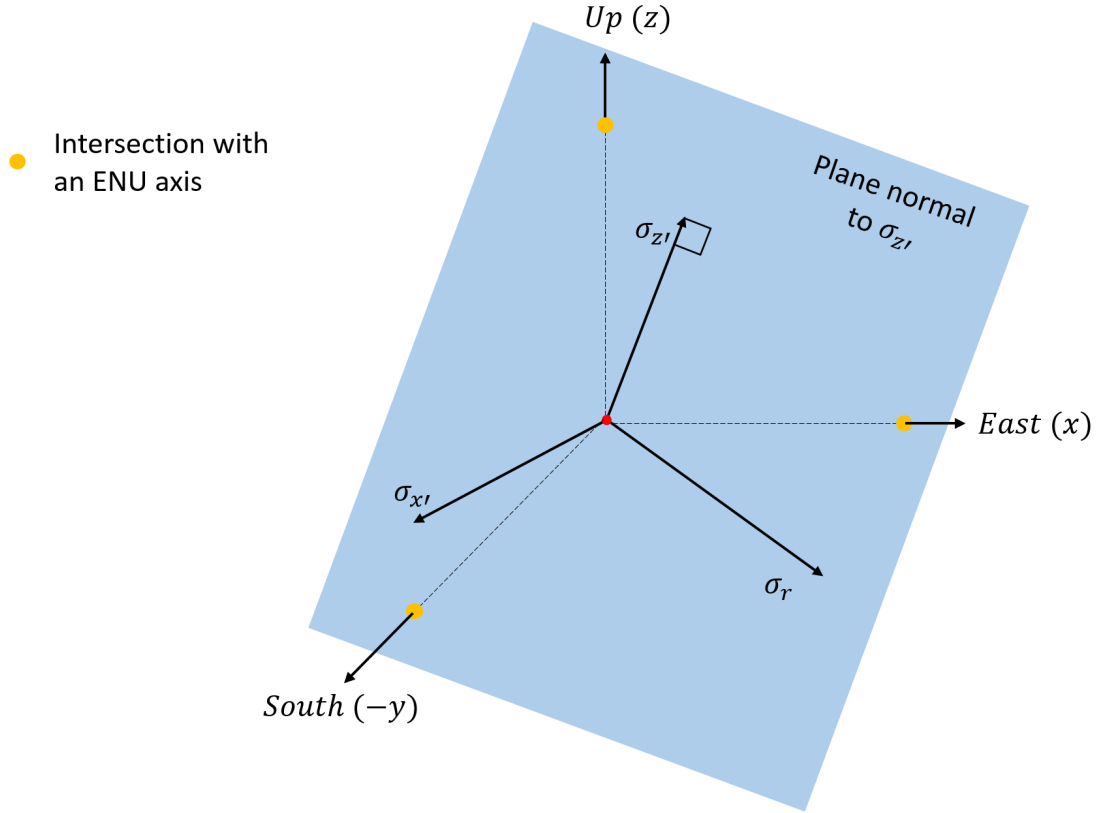


Figure 3.4. The intersections of a plane (perpendicular to a sensor frame covariance axis) with the ENU axes are candidate covariance component approximations

$$\begin{bmatrix} \sigma_{E2} \\ \sigma_{N2} \\ \sigma_{U2} \end{bmatrix} = \begin{bmatrix} \frac{\sigma_{z'}}{\sin \theta \cos \phi} \\ \frac{\sigma_{z'}}{\cos \theta \sin \phi} \\ \frac{\sigma_{z'}}{\cos \theta} \end{bmatrix} \quad (3.18)$$

$$\begin{bmatrix} \sigma_{E3} \\ \sigma_{N3} \\ \sigma_{U3} \end{bmatrix} = \begin{bmatrix} \frac{\sigma_r}{\cos \theta \cos \phi} \\ \frac{\sigma_r}{\cos \theta \sin \phi} \\ \frac{\sigma_r}{\sin \theta} \end{bmatrix} \quad (3.19)$$

The minimum absolute value of each of the candidate covariances is the final compo-

nent of R_{k+1} .

$$\begin{bmatrix} \sigma_E \\ \sigma_N \\ \sigma_U \end{bmatrix} = \begin{bmatrix} \min(\sigma_{E1}, \sigma_{E2}, \sigma_{E3}) \\ \min(\sigma_{N1}, \sigma_{N2}, \sigma_{N3}) \\ \min(\sigma_{U2}, \sigma_{U3}) \end{bmatrix} \quad (3.20)$$

This method operated successfully with or without a finite σ_r . An example of the complete conversion using this method is shown in Figure 3.5. Unrealistically large values for the covariance inputs were used to generate this graphic (for the sake of readability). This method of approximation was implemented as part of the function `LOB2xyzWithCov` shown in Appendix C.

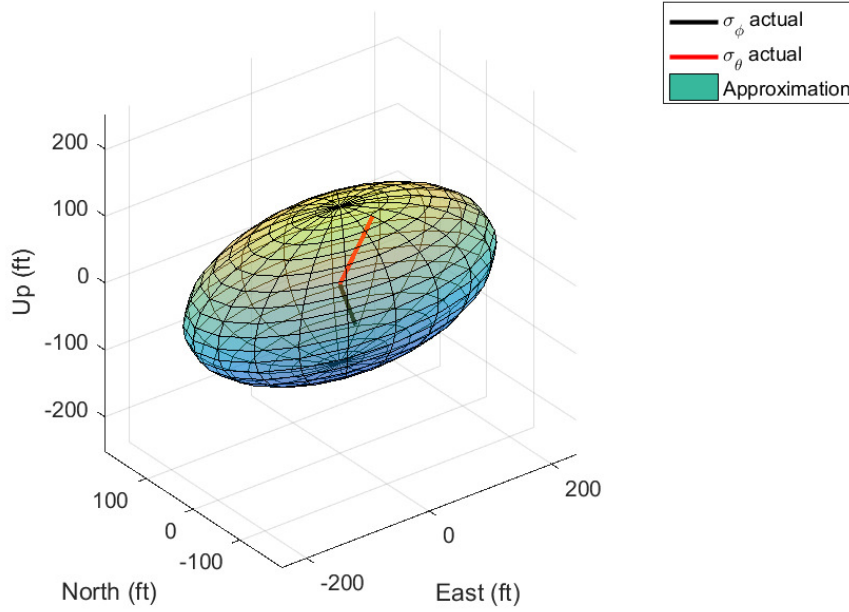


Figure 3.5. The sensor frame covariances approximated as an ENU-aligned ellipsoid

This approximation method converted the LOB measurements and covariances into linearized inputs and sensor covariances for the Kalman filter.

3.6 Choosing a Constant Altitude for Optimization

For the sake of endurance, operating at a fixed altitude was assumed to be a mission requirement. While the ability to drop altitude might decrease the time required to achieve the targeting goal for a single target, the changes in altitude will draw heavily on the UAV's battery or fuel in a mission with multiple targets, ground features, or obstructions. Furthermore, the inclusion of limitations including detection range, cloud cover altitude, and Digital Terrain Elevation Data (DTED) limited the viable altitudes for the UAV to be effective.

Potential cloud cover also places a ceiling on the possible altitudes. Because obstructions to line-of-sight were not considered in this model, cloud cover was assumed to be either overcast or insignificant. Another limitation on maximum altitude for the UAV was the aircraft's flight ceiling. In flat terrain, minimum flight altitude considered was 500 feet AGL. Non-flat terrain, 500 feet above the highest DTED altitude in the neighborhood of operation was the minimum flight altitude. In military operation, a lower AGL altitude buffer could be utilized. To handle vertical features in the DTED (cell towers, buildings, etc.), a No-Fly-Zone (NFZ) was imposed around the highest features and the minimum flight altitude based off of the next highest feature. These altitude limitations are quantified in Equations 3.21 and 3.22. Note that h represents the range of possible altitudes for operation.

$$h \leq \min(h_{vehicle\ max}, h_{cloud\ cover}) \quad (3.21)$$

$$h \geq 500 + \max(DTED_{neighborhood}) \quad (3.22)$$

If detection avoidance is a mission requirement, a more complex influence on altitude results from the audible and visual detection ranges of the UAV. For sim-

plification, both audible and visual detection ranges are approximated as a spherical NFZ. Effectively, there is a sphere placed around the ground target of radius r_{detect} that the UAV can not enter. This is not a direct restriction on the altitude, but influences the selection of viable trajectory altitudes. The value of r_{detect} is selected by the operator.

Similarly, the maximum range between the UAV and target at which the covariance requirement could be attained, $r_{max\ loiter}$, during an orbit influences the altitude selection. As a result, there are two concentric volumes: the inner is a hard constraint within which the mission fails due to detection, the outer is a maximum range for taking useful measurements. The shape and size of this outer volume can be determined through simulation.

Once a fixed altitude is selected, the optimization algorithm can be run. An important result of fixing the altitude is a reduction in the length of the optimized state vector. A smaller state vector generally decreases the time of calculation by orders of magnitude. The only situations significantly impaired by a fixed altitude trajectory are located near or in extreme terrain such as steep mountain sides, small valleys, or urban canyons. These are a separate class of problem not addressed within the scope of this research.

3.7 Direct Method of Path Optimization

Once the problem was fully described and each of the necessary algorithm sub-components designed, a nonlinear programming solver was selected. A direct method was used for this optimization. MATLAB's `fmincon` solver finds the minimum of a constrained nonlinear multivariable function [19]. The inputs of `fmincon` include a cost function, initial guess, linear inequality and equality constraints, upper and lower bounds for the state vector, nonlinear constraints, and options to control the

solver and its outputs. If `fmincon` converged to a plausible solution, the output of the solver was the optimized state vector and cost function value. The formulation of each of these components is described in further detail below.

3.7.1 Cost Function

The scalar cost function maps any multivariate trajectory into a single number. If this problem is framed to minimize the final target covariance in a fixed amount of time given an initial position, the cost function would be a measure of the final covariance P_{final} . The cost function could be the largest diagonal element of P_{final} , the L^2 (or “Euclidean”) norm, or the largest singular value. However, if the problem is to minimize the time required to achieve a covariance goal, the cost function is simply t_f . The min-final-time cost function is simple to implement if the state vector has constant length (a fixed number of unknowns), with final time t_f and time step dt scaling in a continuous fashion. However, if a system with a fixed sensor sampling rate *SensorRate* is simulated, t_f and dt can only scale in a discrete fashion. For these cases, the fixed-time-step min-final-time problem does not converge to an optimal solution. In this situation, the `fmincon` solver “stopped because the size of the current step is less than the default value of the step size tolerance and constraints are satisfied to within the default value of the constraint tolerance” [19].

Because of this issue, the fixed-time-step model justifies the use of bootstrapping to solve as described in Section 3.8. This bootstrapping converts the problem from a min-final-time optimization to several concurrent min-final-covariance problems. The remainder of this section focuses on the framing of a single min-final-covariance problem. The actual function used to evaluate the cost function for any candidate trajectory is attached in Appendix A. The cost function for a min-final-covariance problem is expressed below in Equation 3.23 where P_{final} is assumed to be a diagonal

matrix. The maximum diagonal element of P_{final} is the largest component of the covariance ellipsoid expressed in the ENU coordinate system.

$$J(\bar{x}) = \max_i \left(P_{final_{ii}} \right) \quad (3.23)$$

In order to frame the overall problem as more than a min-final-time, a more comprehensive cost function was developed and is illustrated in Figure 3.6. Instead of minimizing the covariance at the last time step, the covariance at an intermediate time step, t_1 , is minimized. This allows for a two-phase optimization. From t_0 to t_1 , the goal of the UAV is to measure the target covariance with uncertainty below $CovReq$. From t_1 to t_f , the goal of the UAV is to enter a desired orbit of the ground target while maintaining a low target covariance. As well, a weighting on control effort for each time step is used to decrease the intensity of maneuvers. All variables used to calculate the cost function are calculated given the state vector $\bar{x}$ using the functions `getPathFromPsi_d`, `GetLOB`, `LOB2xyzWithCov`, and `StepDKF` (all are included in the appendices and explained throughout Chapter III). The four components of the comprehensive cost function are defined in Equations 3.24 through 3.27. The index where $t = t_1$ is N_{t_1} .

$$f_{t1} = \max_i \left(P(N_{t_1})_{ii} \right) \quad (3.24)$$

$$f_{ctrl} = \sum_{k=1}^N |u(k)| \quad (3.25)$$

$$f_{t2_{cov}} = \sum_{k=N_{t_1}}^N \max_i \left(P(k)_{ii} \right) \quad (3.26)$$

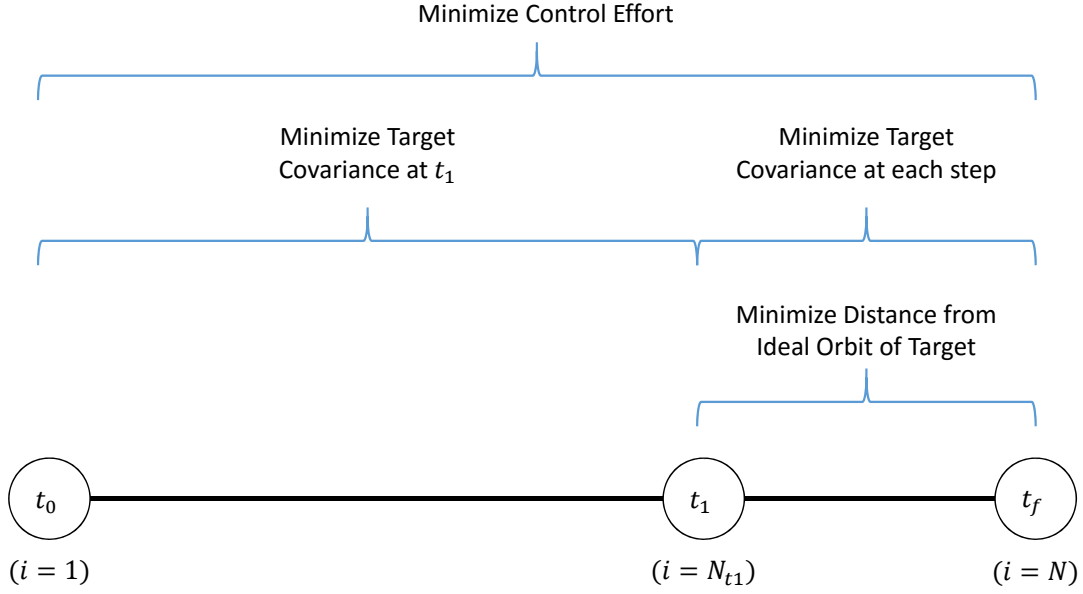


Figure 3.6. The cost function weights control effort, target covariance, and distance from the ideal orbit

$$f_{t2_orb} = \sum_{k=N_{t1}}^N \left| IdealOrbRad - \sqrt{(x(k) - x_t)^2 + (y(k) - y_t)^2} \right| \quad (3.27)$$

The four components of the cost function are combined in Equation 3.28 to include operator-tuneable weights.

$$J = w_1 \times f_{t1} + w_2 \times f_{t2_cov} + w_3 \times f_{t2_orb} + w_4 \times f_{ctrl} \quad (3.28)$$

By setting any combination of the weights to zero, simpler cost functions can be used during simulation. The weight tuning method and results are discussed in Section 4.4. The calculation of the cost function is implemented in the function `CostFun_Min_T1Cov_Orb` attached in Appendix A.

3.7.2 State Vector Formulation

Several options for the formulation of the state vector were considered. Each option required discretizing the state vector from an initial time, $t_0 = 0$, to a final time, t_f , by the fixed time step, dt . Thus, there were $N = \frac{t_f - t_0}{dt}$ time steps for the state vector to store. The first method stored each time step's position (without altitude), heading angle, and heading rate (x , y , ψ , and $\dot{\psi}$ respectively). The resulting state vector was of length $4N + a$, where a represents a small number of additional variables appended to the state vector such as t_f and final covariance. With this state vector, heading angle rate was the control, u , and x , y , and ψ were enforced by constraints based on system dynamics at each time step.

Due to exorbitant run times of the optimization algorithm (hours to optimize a single scenario), a reduced state vector was developed. Only the control, $u(k)$, was included in the state vector for each time step, resulting in a state vector of length $N + a$. The initial position and heading were stored as x_0 . Instead of tracking the position and heading of the vehicle at each time step, x , y , and ψ were calculated forward based off of the state vector $\vec{x}$, UAV airspeed V , wind speed V_w , wind direction ψ_w , and UAV initial condition x_0 using Equations 3.29 through 3.31 for $k = 1$ to $k = N - 1$.

$$\psi_{k+1} = \psi_k + dt u_k \quad (3.29)$$

$$x_{k+1} = x_k + V dt \cos(\psi_k) + V_w dt \cos(\psi_w) \quad (3.30)$$

$$y_{k+1} = y_k + V dt \sin(\psi_k) + V_w dt \sin(\psi_w) \quad (3.31)$$

Calculating the UAV trajectory based off of the $N + a$ control vector is faster than optimizing a full $4N + a$ state vector, and is suitable for the fixed time step format of

this problem. This method of trajectory calculation is implemented in the function `getPathFromPsi_d` attached in Appendix F.

3.7.3 Constraints

The system dynamics and NFZs are enforced as constraints for the `glsnlp` solver. Because the simplified state vector consists only of the control, u , the point-mass model's dynamics are enforced by constraining the control vector in Equation 3.32 for each time step k .

$$-u_{max} \leq u(k) \leq u_{max} \quad (3.32)$$

These constraints on the control, u , are implemented in $2N$ inequalities or instead by using `fmincon`'s built in lower and upper bound arguments.

The NFZ constraints are enforced at every time step. Each NFZ is approximated as a circle. Dynamic NFZs are described with parametric equations for the location of the NFZ center, $x_{NFZ}(k)$ and $y_{NFZ}(k)$, and the radius $R_{NFZ}(k)$. Sampling of a continuous trajectory for the NFZ can discretize any of these values from $f(t)$ to $f(k)$, and interpolation can convert back if required. The inequality constraint for a dynamic NFZ is expressed in Equation 3.33 where $x(k)$ and $y(k)$ were the East and North coordinates of the vehicle at each time step k . Static NFZs are modeled using the same equation with constant values for $R_{NFZ}(k)$, $x_{NFZ}(k)$, and $y_{NFZ}(k)$.

$$g(k) \leq R_{NFZ}(k) - \sqrt{(x(k) - x_{NFZ}(k))^2 + (y(k) - y_{NFZ}(k))^2} \quad (3.33)$$

For the constraint to be enforced, $g(k) \leq 0 \forall k$. In Equation 3.33, $-g(k)$ is the distance from the sensor to the NFZ. Plotting this distance at each time step helps verify the validity of found trajectories.

Multiple NFZs were enforced by repeating Equation 3.33 for each NFZ. The total

number of constraint equations for each optimization was jN , where j was the number of NFZs and N was the number of time steps. The constraints are evaluated in the function `Constraints_NFZ` attached in Appendix B.

3.7.4 Gradients and Hessians

Given the complexity of an iterative Kalman filter with the LOB measurements in this scenario, the gradient and Hessian matrices necessary for the Nonlinear Programming (NLP) solver were calculated numerically within the MATLAB `fmincon` function.

3.8 Bootstrapped Direct Method of Path Optimization

As mentioned in Section 3.7.1, a discrete, fixed time step minimum-final-time optimization did not converge to a true solution in MATLAB's `fmincon` because the cost function (final time) was too insensitive to small steps in the algorithm. An alternative, bootstrapped method solves the same problem using a different cost function while avoiding the convergence issues. Essentially, a min-final-covariance cost function is implemented for an arbitrary final time. If that final covariance is larger than the mission requirement, the final time is extended (either by a fixed number of time steps or by a percentage) and the optimization solved again. If the final covariance is smaller than required, the final time is trimmed and solved again. This process repeats until the covariance goal is reached at the smallest possible final time. Finally, there is a check to determine if the $N_{optimal} - 1$ time step is a possible solution (if final time was extended by a fixed percentage). Figure 3.7 presents a flow chart of this bootstrapping method. The function `Opt_Bootstrapper` attached in Appendix E contains the implementation of this method. Several additional algorithm paths internal to `fmincon` are not included in the flow chart. If the number of computations

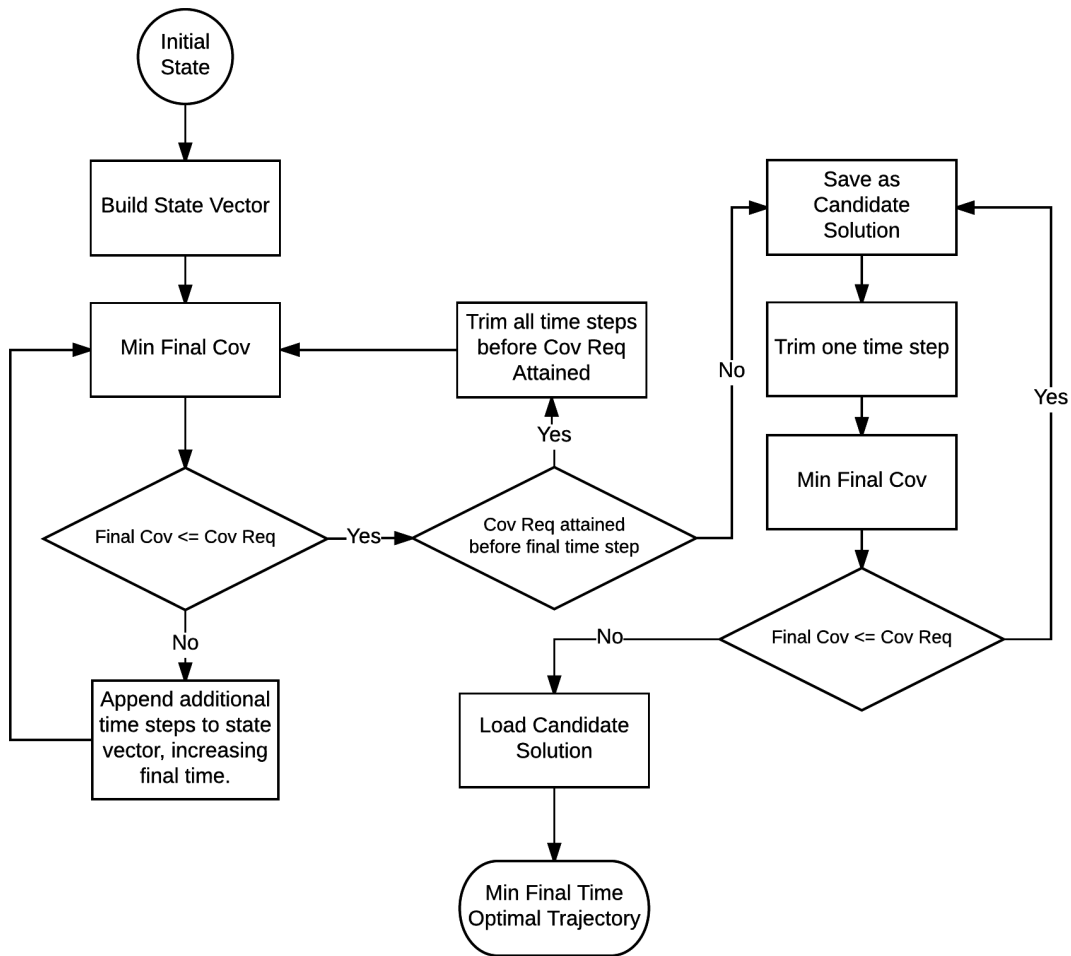


Figure 3.7. The bootstrapping algorithm as a system flow chart

risks too high or no valid solution is found (due to unavoidable constraint violation or an unachievable final covariance requirement), the algorithm stops and outputs its intermediate solution.

This bootstrapped method is an iterative approach using min-final-covariance to approach the min-final-time solution. It avoids algorithm convergence issues due to the fixed time step without becoming computationally inefficient. The top-level flow chart shown in Figure 3.8 demonstrates how the various sub-components fit together in the overall optimization algorithm.

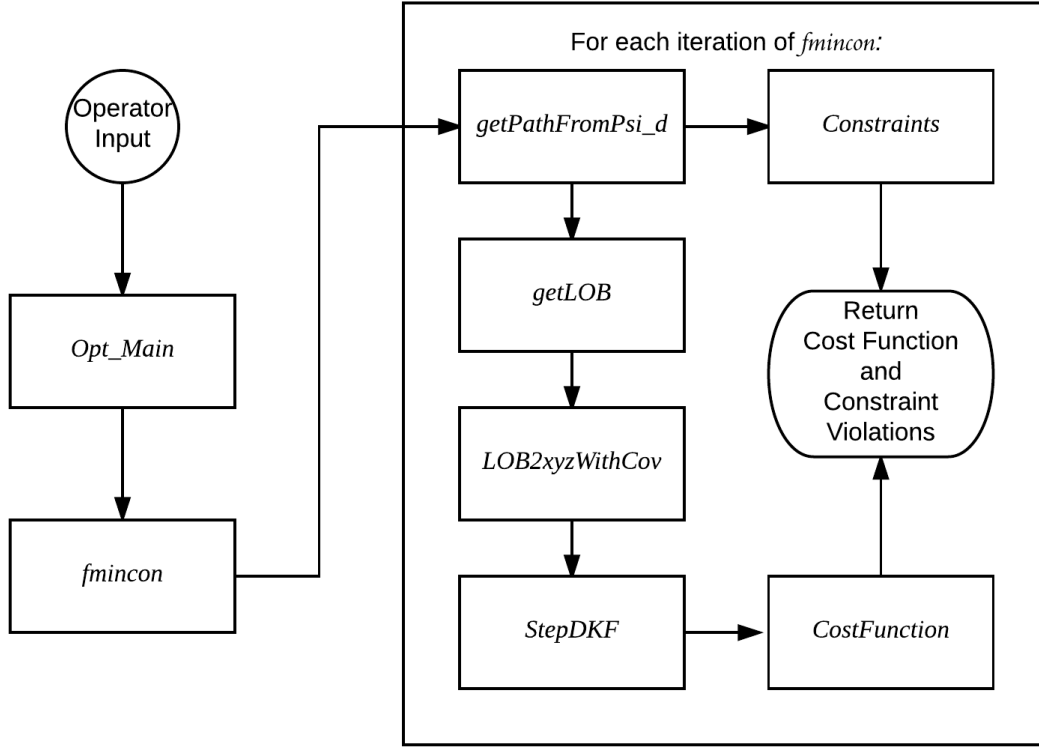


Figure 3.8. The interactions of Opt_Bootstrapper, fmincon, and the other sub-functions

3.9 Other Calculations

Several other components of the algorithm are determined based on the problem's geometry, physics, and statistics. For instance, the limit on turn rate required for the script is needed in units of radians per second. However, the operational turn limit is provided as a maximum acceptable bank angle for the UAV (defined in Table 4.1). The actual turn rate limit is evaluated according to Equation 3.34 (which was derived by a static analysis). In Equation 3.34, g is the acceleration due to gravity, V is the airspeed of the UAV, and ϕ_{max} is the maximum acceptable bank angle.

$$u_{max} = \dot{\psi}_{max} = \frac{V}{g \tan \phi_{max}} \quad (3.34)$$

Another minor component of the problem setup is the specification of NFZ trajectories and sizes. Two NFZs used in the scenarios are based purely on problem geometry: a spherical “detection” NFZ centered on the target and a conical dynamic NFZ due to a secondary air asset sensor’s field of view. The detection NFZ location was centered on the target, and the radius is determined using Equation 3.35. *SlantRangeMin* was a scenario-specific value and was defined in Table 4.1. The altitude of the UAV is z_v . Note that many of the variables are named to coincide with their values in Table 4.1 and in the sample MATLAB code provided in the Appendices.

$$R_{detect} = \sqrt{SlantRangeMin^2 - z_v^2} \quad (3.35)$$

The dynamic NFZ is determined based on the trajectory and characteristics of the secondary aircraft. The geometry of the dynamic NFZ is shown in Figure 3.9. The intersection of the cone with the altitude level of the UAV is approximated as a circle. The radius of the NFZ circle is defined in Equation 3.36. The circular NFZ trajectory has a constant radius because all scenarios involving the dynamic NFZ assume the secondary aircraft to be orbiting the target. The half-angle of the cone is *NFZ_Spread*. The target-orbiting aircraft is designated *AC*.

$$R_{dyn\ NFZ} = \frac{(AC_Alt - z_v)}{2} \left(\sin(AC_LookAngle + NFZ_Spread) - \sin(AC_LookAngle - NFZ_Spread) \right) \quad (3.36)$$

Equation 3.37 defines *AC_NFZ_Rad*, the radius of the dynamic NFZ’s orbit. The East and North components of the dynamic NFZ’s position are defined in Equations 3.38 and 3.39. *NFZ_2_theta* is the vector describing the angular position of the aircraft

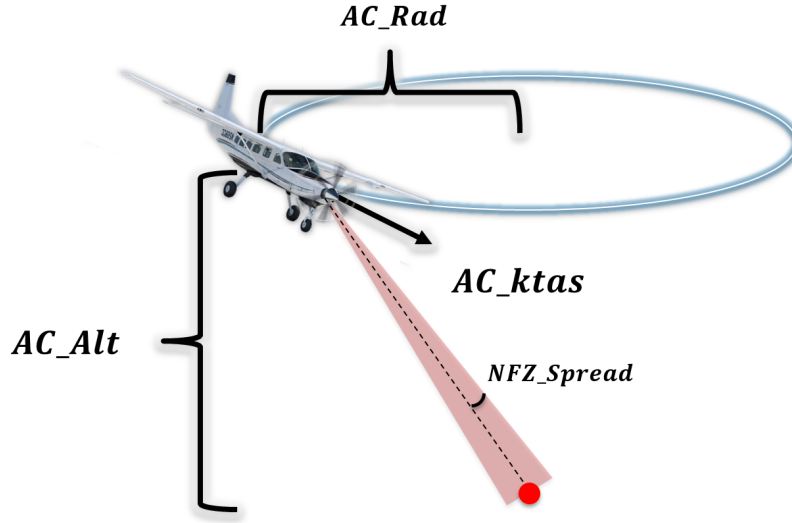


Figure 3.9. The geometry of the dynamic NFZ due to a target-orbiting aircraft

about the target at every time step.

$$AC_NFZ_Rad = AC_Rad - (AC_Alt - z_v) \sin \left(\arctan \left(\frac{AC_Rad}{AC_Alt} \right) \right) \quad (3.37)$$

$$X_{NFZ_i} = \cos(NFZ_2_theta_i) AC_NFZ_Rad + X_t(1) \quad (3.38)$$

$$Y_{NFZ_i} = \sin(NFZ_2_theta_i) AC_NFZ_Rad + X_t(2) \quad (3.39)$$

The target covariance requirement $CovReq$ is one of the most crucial values for the min-final-time geolocation problem. The first time step when the maximum component of the target covariance is decreased to less than $CovReq$ is t_1 . The actual value is chosen to meet a 6 meter Circular Error Probable (CEP) 90 requirement. CEP 50 was well defined in a DTIC report by William Nelson: “The CEP associated with an estimated target location (x,y) is defined to be the radius of the smallest

circle with center at (x,y) which has a 50% probability of containing the true target coordinates” [20]. CEP 90 is similarly the smallest circle with a 90% probability of containing the true target coordinates. By assuming a zero-mean Gaussian distribution [20], the 90% confidence level was also defined using a commonly known statistical value, $Z_{\frac{\alpha}{2}}$. For a 90% confidence interval, $Z_{\frac{\alpha}{2}} = 1.645$. Accordingly, $CovReq$ is calculated using Equation 3.40.

$$CovReq = \frac{6[m]3.28084[\frac{ft}{m}]}{Z_{\frac{\alpha}{2}}} = 11.9666[ft] \quad (3.40)$$

Thus, once the largest component of the target covariance is decreased to less than 11.9666 feet, the Kalman filter achieves a 90% confidence measurement of the target location within a 6 meter radius circle.

3.10 Optimization Problem Setup

The scalar control, u , is the heading angle rate, $\dot{\psi}$, from $t = t_0 \rightarrow t_1 \rightarrow t_f$, which in discretized form is the vector of control signals. The states x , y , and covariance matrix P are generated given the time history of u by the functions `getPathFromPsi_d`, `getLOB`, `LOB2xyzWithCov`, and `StepDKF`.

3.10.1 Continuous Setup

To solve an optimization problem, the cost function, dynamic constraints, system dynamics, and boundary conditions must be defined. These are set forth in Equations 3.41 through 3.45.

Minimize the cost function with t_0 , t_1 , and t_f specified

$$J(t_1, u) = w_1 \max_i \left(P(t_1)_{ii} \right) + w_2 \int_{t_0}^{t_f} |u(t)| dt + w_3 \int_{t_1}^{t_f} \max_i \left(P(t)_{ii} \right) dt \\ + w_4 \int_{t_1}^{t_f} \left| IdealOrbRad - \sqrt{(x(t) - x_t)^2 + (y(t) - y_t)^2} \right| dt \quad (3.41)$$

subject to the dynamic constraints

$$g(t) = \begin{bmatrix} R_{NFZ,1}(t) - \sqrt{(x(t) - x_{NFZ,1}(t))^2 + (y(t) - y_{NFZ,1}(t))^2} \\ R_{NFZ,2}(t) - \sqrt{(x(t) - x_{NFZ,2}(t))^2 + (y(t) - y_{NFZ,2}(t))^2} \\ R_{NFZ,3}(t) - \sqrt{(x(t) - x_{NFZ,3}(t))^2 + (y(t) - y_{NFZ,3}(t))^2} \\ |u(t)| - u_{max} \end{bmatrix} \leq 0, \quad \forall t \quad (3.42)$$

the state dynamics

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} V \cos(\psi) + V_w \cos(\psi_w) \\ V \sin(\psi) + V_w \sin(\psi_w) \\ u \end{bmatrix}, \quad \forall t \quad (3.43)$$

and the boundary conditions

$$\begin{bmatrix} x(t_0) \\ y(t_0) \\ \psi(t_0) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ \psi_0 \end{bmatrix} \quad (3.44)$$

The bootstrapper minimizes t_1 subject to the boundary condition

$$\max_i \left(P(t_1)_{ii} \right) \leq CovReq \quad (3.45)$$

3.10.2 Discrete Setup

In order to utilize `fmincon` as an NLP, the continuous setup must be discretized.

The discretized optimization problem is presented in Equations 3.46 through 3.50.

Minimize the cost function (where N_{t_1} is the value of k for $t = t_1$)

$$J(N_{t_1}, u) = w_1 \max_i \left(P(N_{t_1})_{ii} \right) + w_2 \sum_{k=1}^N |u(k)| + w_3 \sum_{k=N_{t_1}}^N \max_i \left(P(k)_{ii} \right) \\ + w_4 \sum_{k=N_{t_1}}^N \left| IdealOrbRad - \sqrt{(x(k) - x_t)^2 + (y(k) - y_t)^2} \right| \quad (3.46)$$

subject to the dynamic constraints

$$g(k) = \begin{bmatrix} R_{NFZ,1}(k) - \sqrt{(x(k) - x_{NFZ,1}(k))^2 + (y(k) - y_{NFZ,1}(k))^2} \\ R_{NFZ,2}(k) - \sqrt{(x(k) - x_{NFZ,2}(k))^2 + (y(k) - y_{NFZ,2}(k))^2} \\ R_{NFZ,3}(k) - \sqrt{(x(k) - x_{NFZ,3}(k))^2 + (y(k) - y_{NFZ,3}(k))^2} \\ |u(k)| - u_{max} \end{bmatrix} \leq 0, \quad \forall k \quad (3.47)$$

the state dynamics (where Δt is the time step)

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \psi(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + V \Delta t \cos(\psi(k)) + V_w \Delta t \cos(\psi_w) \\ y(k) + V \Delta t \sin(\psi(k)) + V_w \Delta t \sin(\psi_w) \\ \psi(k) + u \Delta t \end{bmatrix}, \quad \forall k \quad (3.48)$$

and the boundary conditions

$$\begin{bmatrix} x(0) \\ y(0) \\ \psi(0) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ \psi_0 \end{bmatrix} \quad (3.49)$$

The bootstrapper minimizes N_{t_1} subject to the boundary condition

$$\max_i \left(P(N_{t_1})_{ii} \right) \leq CovReq \quad (3.50)$$

3.11 Simulations

Once the bootstrapping algorithm was developed, any set of initial states, covariance requirements, and NFZs could be simulated. The selection of the simulation's initial conditions was based on a variety of real-world situations. Simulations with each of the main simulation settings varied independently can determine their effects on algorithm and mission performance. By using real-world values for $\dot{\psi}_{max}$, V , h , σ_ψ , and σ_θ , predictions on real-world performance can be made. As well, a qualitative analysis of the different simulated trajectories exposes geometric trends in the optimal path. These trends could later be used to form a real-time near-optimal control law for application in onboard processing.

3.12 Summary

The process of taking LOB measurements from any candidate trajectory, geometrically approximating the conversion of covariance to the ENU coordinate frame, and updating a KF to estimate target location and covariance was fully developed. Different versions of the path-optimization algorithm were configured using these components to compare mission and computational performance. The results are presented in the next chapter.

IV. Results

4.1 Chapter Overview

With the methodology and optimization functions fully developed, the only remaining operator decisions were the selection of hard-coded settings and test scenarios. Simulation and optimization settings were determined based off of real-world and calculated values, and test scenarios were developed to prove the optimization algorithm’s capabilities and simulate practical events. Finally, the simulated results provided a basis for predictions in real-world performance and determination of the optimal trajectory’s characteristics.

4.2 Simulation Settings

Many of the settings for the simulations were selected to characterize a Tactical Off-Board Sensing (TOBS) Unmanned Aerial Vehicle (UAV) proposed by AFRL. The proposed values were inputted to the script in US customary units (KTAS, degrees, feet AGL, Nautical Miles, etc.), with unit conversions to common units (feet, feet per second, radians, etc.) to ensure correct scaling. Other simulation settings were calculated using the equations in Section 3.9. All of the user-specified variables for the script and their values were set forth in Table 4.1. Some settings were variable based purely on the operator’s preference and are customizable for specific scenarios. As well, the options used within MATLAB’s `fmincon` for most simulations were recorded in Table 4.2.

In several trials, the algorithm used by `fmincon` was changed to ‘sqp’, an optimization algorithm which maintains feasibility with respect to the bounds [19]. This algorithm ran faster than ‘interior-point’ in most cases, but occasionally lost feasibility within the bounds due to constraint violations. This issue may have been due to

inequality constraints containing an absolute value function. Also, ‘StepTolerance’ was decreased to 10^{-12} for some solutions requiring a longer state vector.

Table 4.1. Algorithm Variables, Values, and Descriptions

Variable	Value	Source	Description
V	55 (KTAS)	Vehicle	Constant air speed of UAV
$\dot{\psi}_{max}$	30 ($^{\circ}$)	Vehicle	Maximum bank angle
ψ_{max}	XXX (rad/sec)	Calculated	Maximum turn rate
t_f	Variable (sec)	Operator	Initial guess for final time
t_2	30 (sec)	Scenario	Fixed time after CovReq attained before simulation ends
ExtendRatio	1.25	Operator	Ratio used to extend state vector during bootstrapping
$X_o(1)$	-1.5 (NM)	Scenario	Initial UAV East coordinate
$X_o(2)$	0.25 (NM)	Scenario	Initial UAV North coordinate
$X_o(3)$	$\frac{\pi}{4}$ (rad)	Scenario	Initial UAV heading
z_v	500 (ft AGL)	Scenario	UAV Altitude
SlantRangeMin	1500 (ft)	Scenario	Minimum slant range from UAV to target (for Detection NFZ)
IdealSlantRange	2000 (ft)	Scenario	Ideal slant range from UAV to target
IdealOrbRad	Variable (ft)	Calculated	Ideal UAV orbit radius
SensorRate	1 Hz	Operator	Rate of LOB measurements
Sv(1,1)	0.00275 (rad)	Scenario	Sensor covariance (azimuth)
Sv(2,2)	0.00075 (rad)	Scenario	Sensor covariance (elevation)
Sv(3,3)	30 (ft)	Scenario	Sensor covariance (range)
Q_k	$I_{3 \times 3}(ft)$	Scenario	System noise (low for stationary targets)
AC_IC_Ang	π (rad)	Scenario	Initial angular position of dynamic NFZ source
AC_Alt	10000 (ft)	Scenario	Altitude of dynamic NFZ source
AC_Rad	6600 (ft)	Scenario	Orbit radius of dynamic NFZ source
AC_KTAS	220 (KTAS)	Scenario	Constant ground speed of dynamic NFZ source
NFZ_SpreadDeg	5($^{\circ}$)	Scenario	Half-angle of dynamic NFZ cone
X_t	[0, 0, 0] (ft)	Scenario	Actual position of target in ENU coordinates
Z_alpha2_90	1.645	Scenario	Alpha level for a 90% confidence interval
CovReq	11.9666 (ft)	Calculated	Maximum allowed component magnitude for target covariance at t_1
WindDir	Variable (rad)	Scenario	Constant wind direction
WindSpd	0 or 15 (KTAS)	Scenario	Constant wind speed

Table 4.2. Specification of MATLAB `fmincon` Options

Setting	Value	Description
Algorithm	‘interior-point’	MATLAB’s robust default optimization algorithm that maintains ”feasibility“ at all iterations [19]
HessianApproximation	‘bfgs’	Calculates the Hessian by a dense quasi-Newton approximation [19]
ScaleProblem	‘obj-and-constr’	Causes the algorithm to normalize all constraints and the objective function [19]
StepTolerance	1e-8	A lower bound on the size of a step which determines when iterations end [19]
OptimalityTolerance	1e-6	A tolerance for the first-order optimality measure [19]
TolCon	1e-6	A termination tolerance for constraint violations [19]

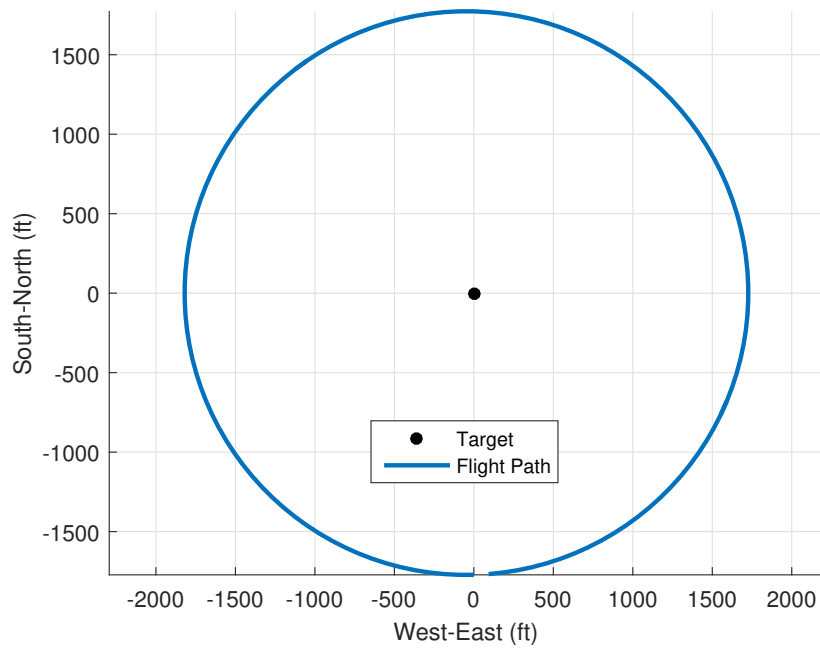
4.3 Simplified Examples

Several simple trajectories (with known results) were simulated to demonstrate and validate the operation of the KF with its linearizing approximations. In Figure 4.1a, a clockwise orbit about the target at 500 ft AGL and at the specified operational slant range *IdealSlantRange* was simulated.

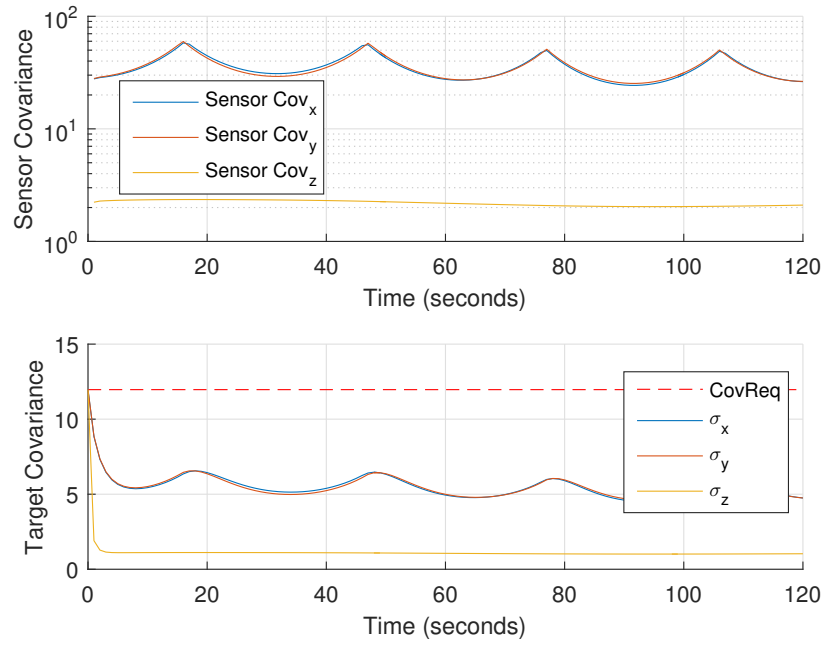
For this example only, the initial target covariance matrix, Q_k , was initialized at $CovReq \times I_3$ instead of $100 \times I_3$ (a high value representing the operator’s uncertainty of the initial target estimate). The sensor covariance section of Figure 4.1b shows the diagonal components of P_{k-k} to the KF. The wave-like shapes of the covariance histories were a result of the linear approximation used to convert from the sensor frame to the ENU coordinate frame.

In Figure 4.1b, the values for each component of the sensor and target covariance in the ENU coordinate system are displayed. When σ_x , σ_y , and σ_z were all below *CovReq*, the targetting requirement was met. Thus, in the majority of the covariance plots, only the maximum component of σ_x , σ_y , and σ_z are displayed.

In Figure 4.2a, four flyby trajectories at 500 feet AGL and different offsets were simulated. The green diamonds represent the first point along the trajectory at which the required target covariance is achieved. The red diamonds represent the last measurement before the target covariance rose above *CovReq*. The sensor and target covariance histories were shown in Figure 4.2b. The spike in covariance in Figure 4.2b was due to a discontinuity in the Z-axis component of sensor covariance (from Equation 3.19). Because no range sensor was modeled, σ_r was set to ∞ or an arbitrarily high value. As the elevation angle approached 90° , the sensor covariance in the Z-axis approached infinity (unless a range sensor was utilized). This “flyover” spike was avoided in other trials by the enforcement of the Detection No-Fly-Zone



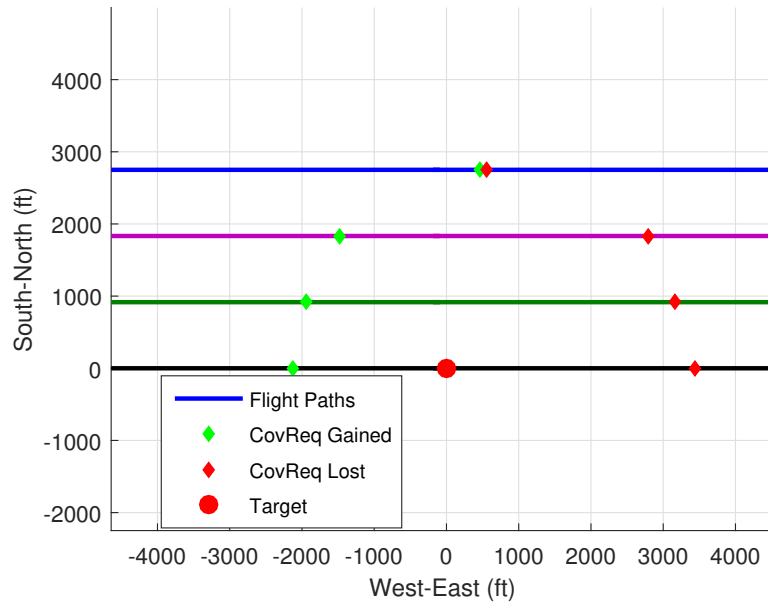
(a) The clockwise orbit trajectory



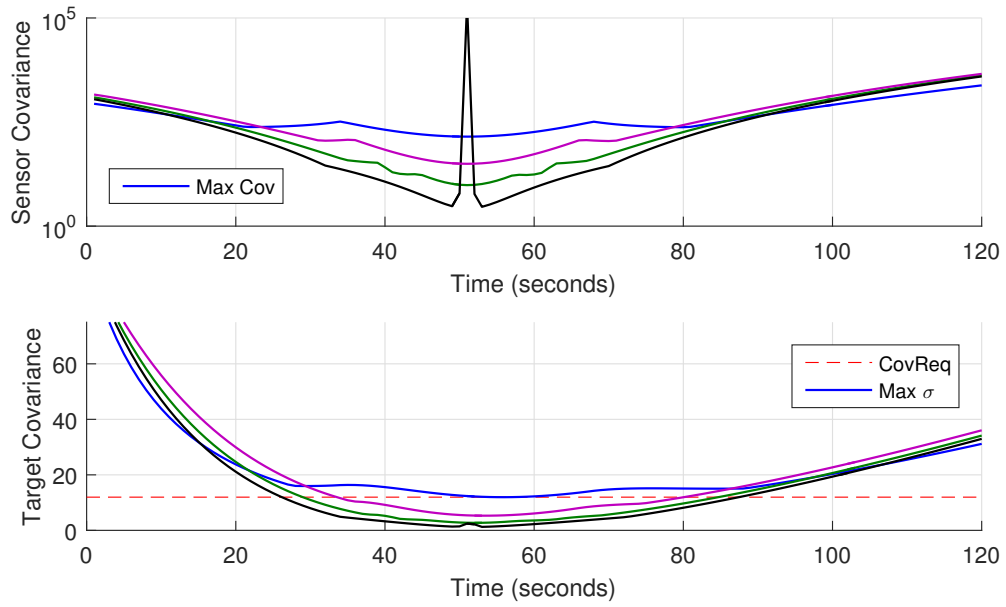
(b) Each component of the sensor and target covariances with respect to time

Figure 4.1. A clockwise orbit of the target at 500 ft AGL and slant range of 2000 ft

(NFZ) centered on the target. An additional way of avoiding the flyover spike would be to set a smaller value for σ_r or model a range sensor with a known uncertainty.



(a) The trajectories and the points where CEP90% is gained and lost



(b) The maximum components of sensor and target covariances with respect to time

Figure 4.2. Four west-east flybys simulated at different offsets from the target

4.4 Variation of the Cost Function

The MATLAB code for the cost function used in the optimization algorithm is shown in Appendix A. Weights were placed on four different components. w_1 was the weight on the maximum target covariance at time t_1 . This was the primary weight of the cost function and served to minimize t_1 within the bootstrapping algorithm. w_4 was the weight on the maximum target covariance at each time step after t_1 . w_3 was the weight on distance between the UAV and the ideal orbit radius for each time step after t_1 . w_2 was the weight on control effort at each time step.

Table 4.3 shows the values of the weights tested for each trial shown in Figure 4.3. The magnitudes of each weight (3,000, 10, 2, and 1,000 respectively) were selected by manual tuning by the operator. The weights were scaled to prioritize (from most to least important) w_1 , w_3 , w_4 , and w_2 . The resulting path generated using CostFunc5 was not a pure min-final-time solution, but achieved similar results with less control and a more ideal final state. Rather than flying directly towards the detection NFZ at t_f , the UAV trajectory turns into an orbit about the ground target.

Table 4.3. Variation of Cost Function Weights

Trial	w_1	w_2	w_3	w_4
J_1	3000	0	0	0
J_2	3000	0	0	10
J_3	3000	1000	0	0
J_4	3000	0	2	0
J_5	3000	1000	2	10

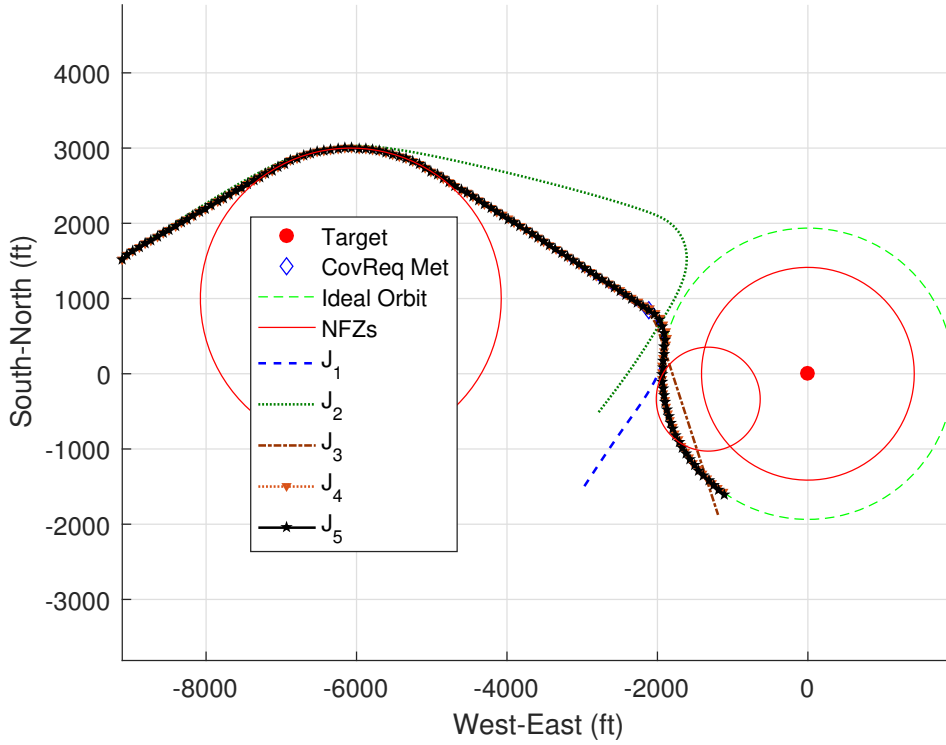


Figure 4.3. Comparison of flight paths for the cost functions outlined in Table 4.3

4.5 Optimal Solution to a Real-World Scenario

A scenario was designed to emulate the approach to a ground target and entry to a steady state orbit typical of a TOBS mission. A fixed flight altitude of 500 ft AGL was selected as a reasonable altitude for a given operation. The simulated UAV was subject to a constant wind due south at 15 KTAS. The time after *CovReq* was attained to enter and maintain the ideal orbit during simulation, t_2 , was 30 seconds. A NFZ was located between the UAV's initial position and the target to model the presence of an arbitrary (but non-measurement-blocking) obstruction. Additionally, a NFZ was centered on the ground target to ensure the UAV maintained an acceptable minimum slant range. Finally, a dynamic NFZ was modeled to simulate the boresight of a sensor on board a cooperative aircraft circling the target at 220 KTAS, 10,000

ft AGL, and with an orbit radius of 6,600 ft. The sensor’s conical field-of-view half-angle was 5° . The dynamic NFZ forced the UAV to avoid the orbiting aircraft sensor’s field-of-view. The problem geometry and orbit end-condition are shown in Figure 4.4. AC_Rad , AC_Alt , and AC_ktas are fully defined in Table 4.1.

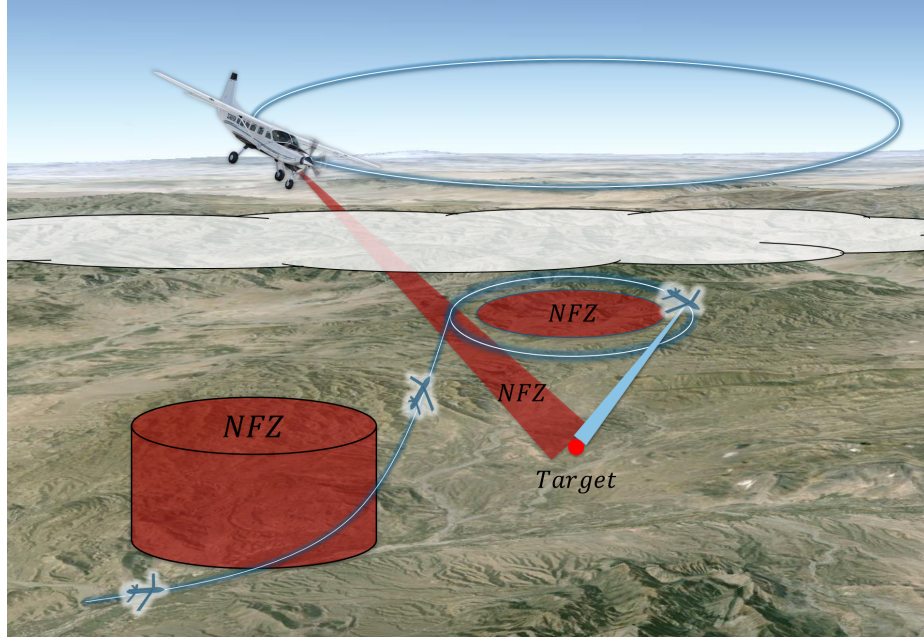
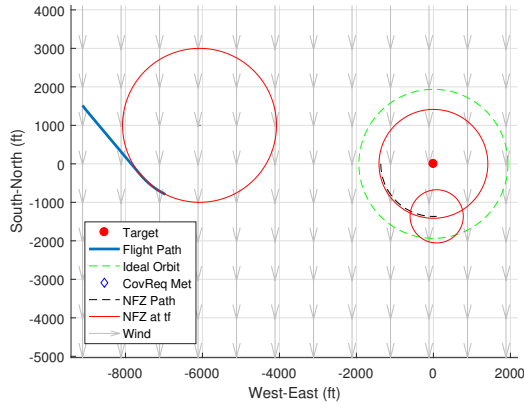


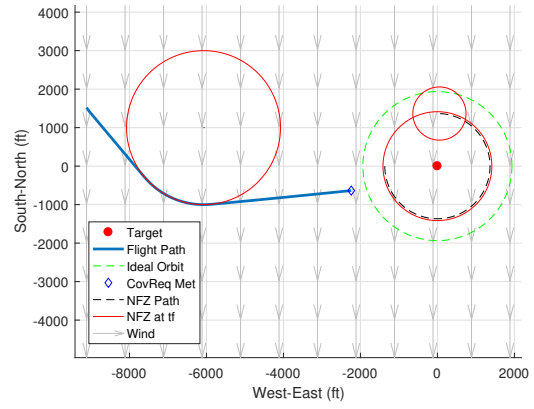
Figure 4.4. The geometry of a real-world scenario

The optimal path of the UAV closely resembles a “Dubin’s Path”: a trajectory consisting only of straight segments and segments of maximum possible turn rate [21]. As shown in Figures 4.6 and 4.7, the control vector deviates from zero when a full turn was specified or when a NFZ constraint was active. However, due to the discretization of the trajectory and the cost function’s weights on control effort and distance from the ideal orbit radius, the solution was not an exact minimum final time Dubin’s path.

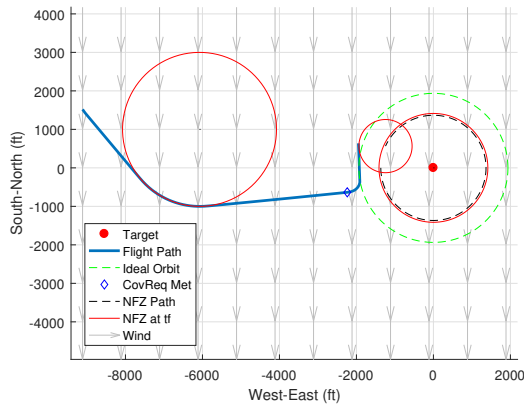
The targeting goal was met at a ground range of 0.38 NM to the target, 85 seconds after launch. Next, with one complete scenario modeled and optimized, the independent effects of varying altitudes, initial conditions, and wind directions were studied.



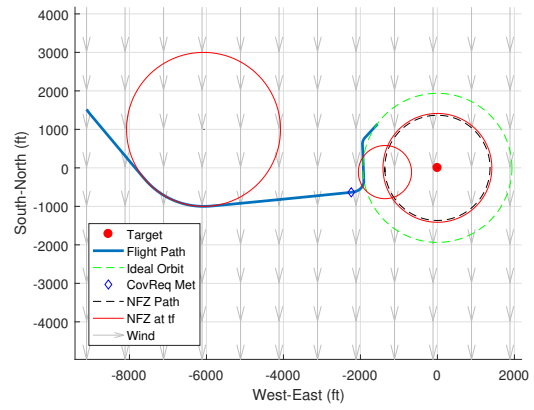
(a) UAV before t_1



(b) UAV at t_1



(c) UAV after t_1



(d) UAV at t_f

Figure 4.5. The NFZs and trajectory of the UAV for a real-world scenario with wind

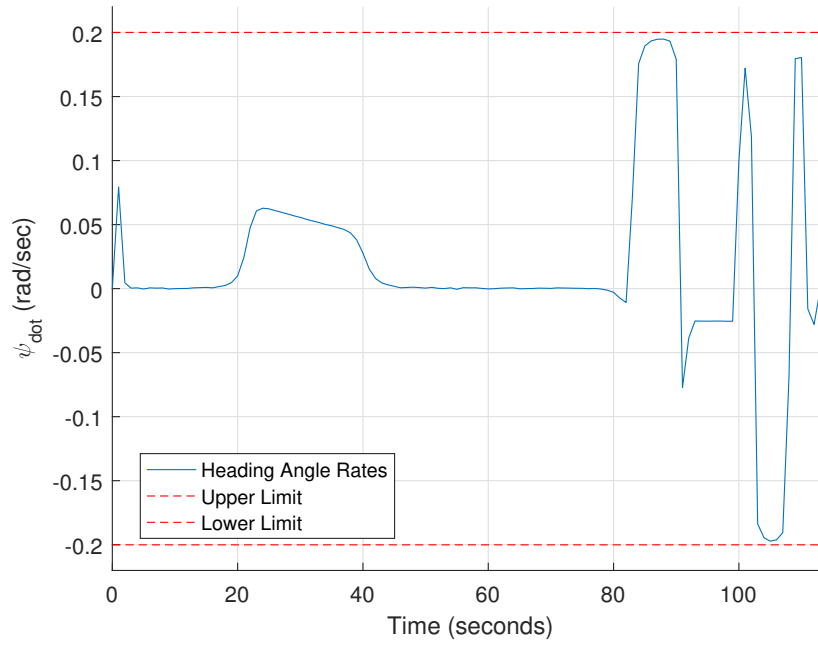


Figure 4.6. The optimal control vector, $\dot{\psi}$, for a real-world scenario with wind

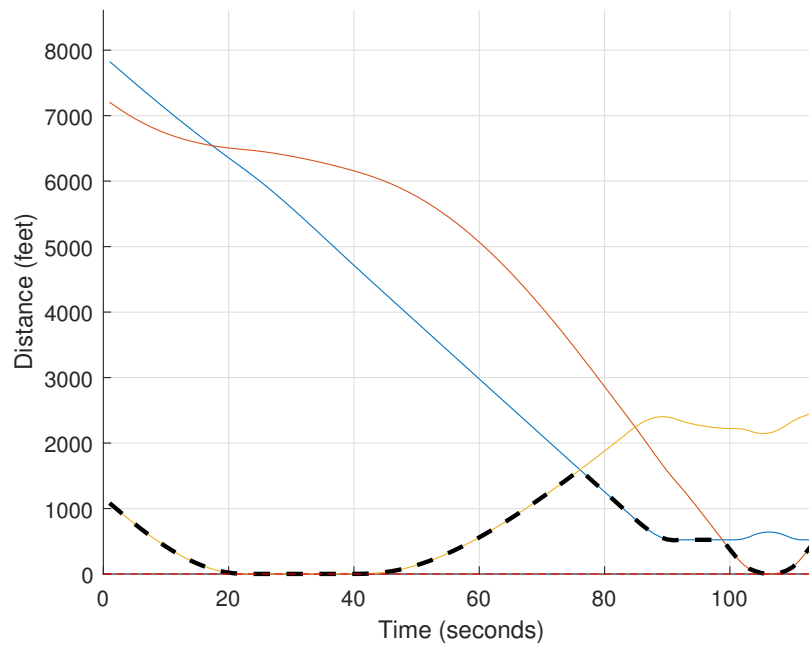


Figure 4.7. The distance to each NFZ with respect to time for a real-world scenario with wind

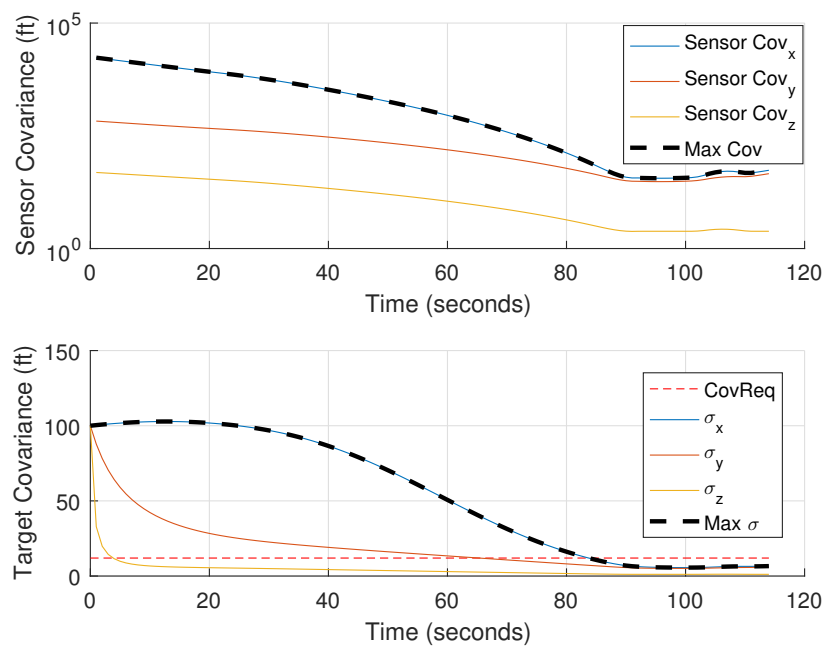


Figure 4.8. The sensor and target covariances with respect to time for a real-world scenario with wind

4.6 Comparison of Different Altitudes

Although altitude was selected to be 500 ft AGL, a comparison of different constant altitudes explored the resulting trajectory changes. The optimal paths at different altitudes are compared in Figure 4.9. The detection and dynamic NFZs were not displayed due to their dependence on UAV altitude. The targeting time, ideal orbit radius, and other relevant values are compiled in Table 4.4.

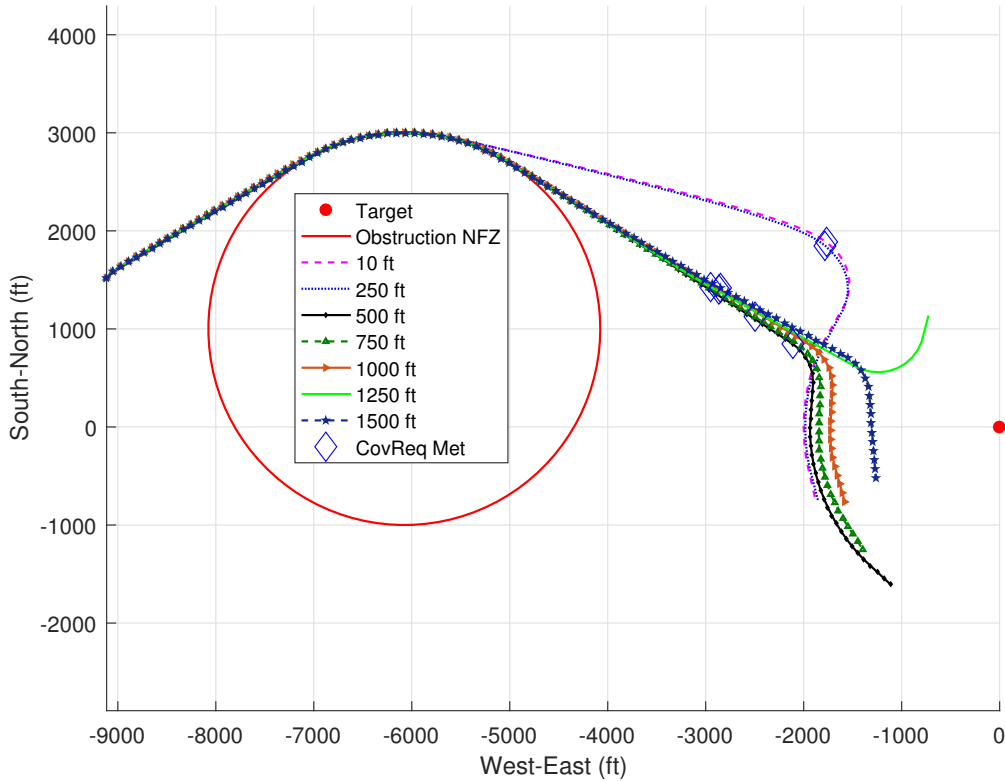


Figure 4.9. Comparison of optimal flight paths at for several altitudes (detection and dynamic NFZs not shown)

Table 4.4 displays a trend that higher altitudes achieve *CovReq* faster and have a smaller detection NFZ to avoid, with 1,250 ft AGL having the minimum time of all simulated trials. The point at which the 1,250 ft AGL trial achieved *CovReq* had an elevation angle to the target of approximately 20° below the horizon and

Table 4.4. The Effects of Different UAV Altitudes

Altitude (ft AGL)	t_1 (sec)	<i>IdealOrbRad</i> (ft)	Detection NFZ Radius (ft)
10	86	2000	1500
250	86	1984	1479
500	87	1937	1414
750	82	1854	1299
1000	77	1732	1118
1250	76	1561	636
1500	77	1323	0

slant range of 3,510 ft (a ground distance to the target of 0.54 NM). As well, that position was relatively close to the diagonal between the East and North axes (19° from the NW diagonal). Geometrically, these results were reasonable given the values of the sensor’s azimuth angle covariance ($Sv(1, 1) = 0.00275$ rad) and elevation angle covariance ($Sv(2, 2) = 0.00075$ rad) and the method of linearizing approximation explained in Section 3.5.

In comparison, when *CovReq* was achieved for the 500 ft AGL trial, the elevation angle to the target was 12° below the horizon, the position was 23° from the NW diagonal, and the slant range was 2,330 ft. Although the sensor covariance at t_1 was lower for the 500 ft AGL trial than 1,250 ft AGL, the measurements immediately before t_1 were much higher. This insight demonstrated both the influence of geometry on sensor covariance and the need for multiple measurements with low covariance to drive the KF’s target covariance below *CovReq*.

Regardless, the actual TOBS mission has several requirements not modeled in this algorithm: primarily the task of identifying the ground target. Although the target would be geolocated faster at higher altitudes, vision based sensors might require a lower altitude or slant range to successfully identify the target. The unexpected curves at the end of the 1,250 ft and 1,500 ft altitude trial flight paths were due to the dynamic NFZ. Both trials required significant deviation from the smaller ideal

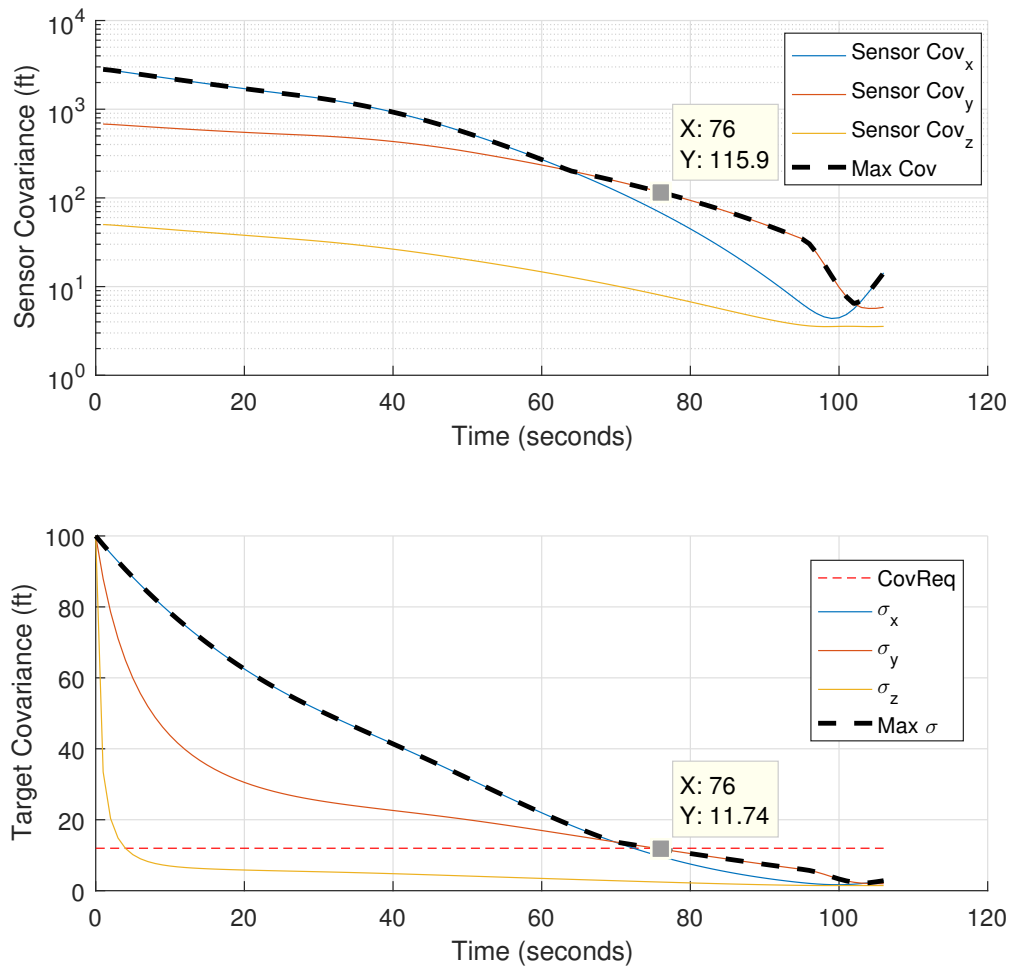


Figure 4.10. The sensor and target covariances with respect to time for a 1,250 ft AGL trajectory

orbit radius due to the timing of the dynamic NFZ. These unexpected curves were not a direct influence of the altitude change.

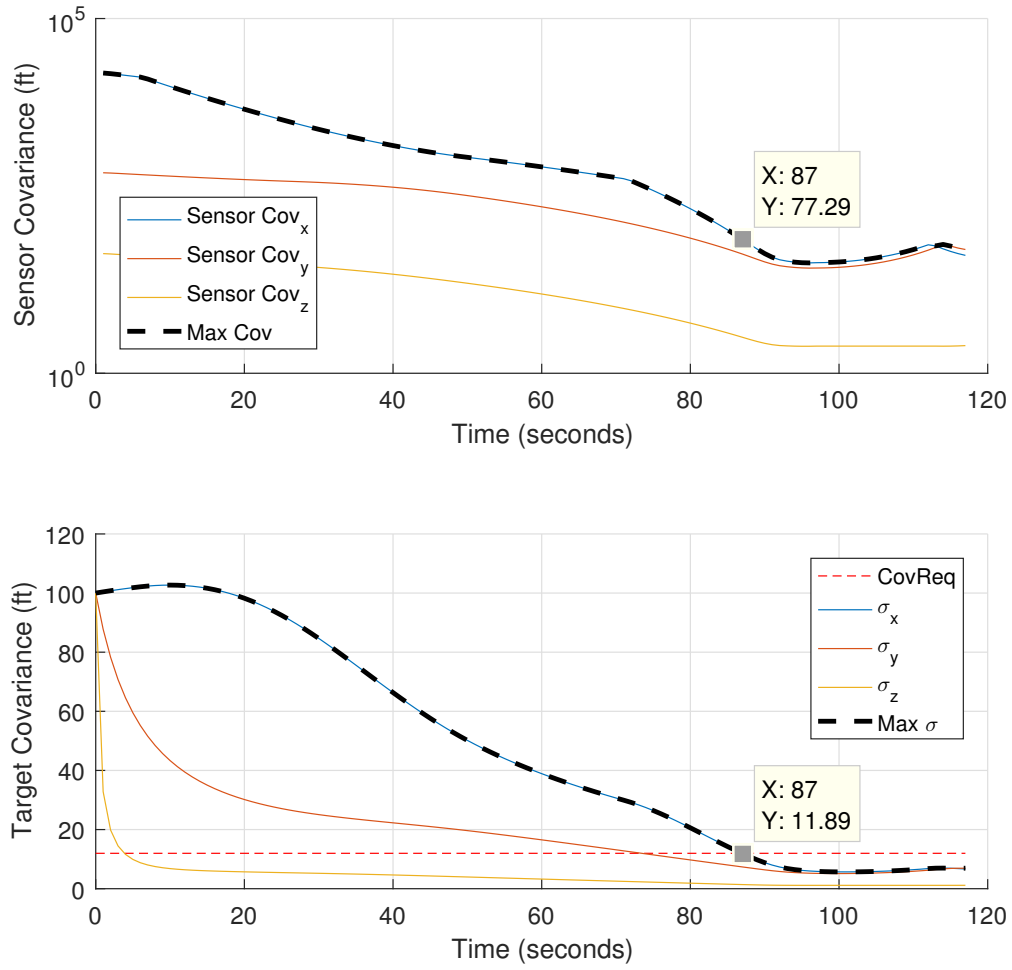


Figure 4.11. The sensor and target covariances with respect to time for a 500 ft AGL trajectory

4.7 Dependence on Initial Conditions

Given the same NFZs as the scenario in Section 4.5, the effects of different initial positions (P) and headings (ψ) on the optimal flight path were compared. The different sets of initial conditions simulated are recorded in Table 4.5 and the paths are compiled in Figure 4.12.

With the exception of the $P_1\psi_1$ trial, each optimal path passed south of the obstruction NFZ. Each of the southern paths approached the target directly until

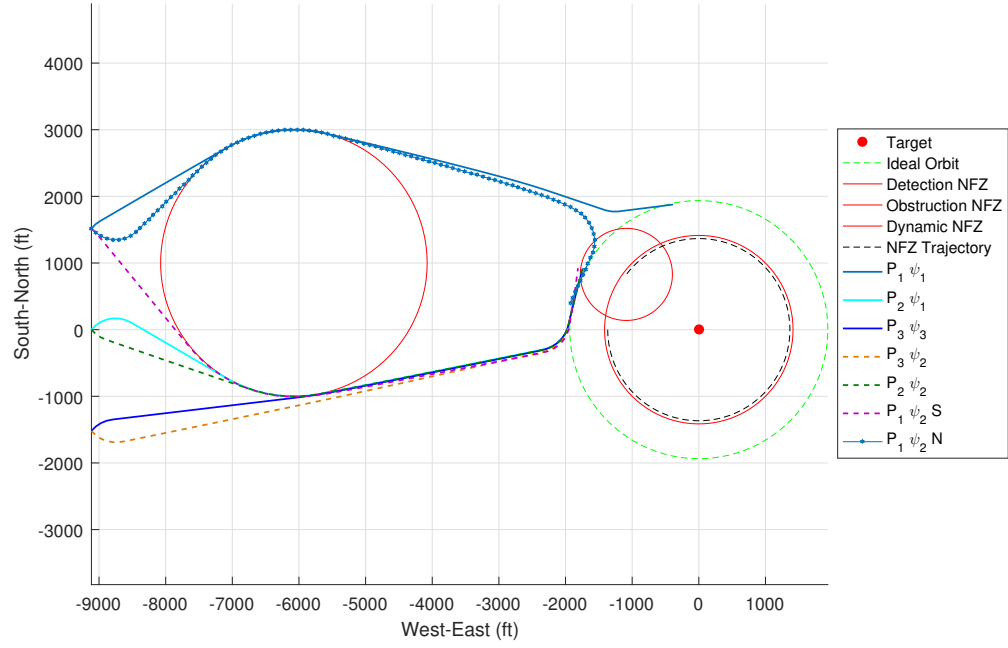


Figure 4.12. Comparison of the NFZs and trajectories for the initial conditions listed in Table 4.5

Table 4.5. Variation of Initial Conditions

Position	East (NM)	North (NM)	ψ	Heading (rad)	t_1 (sec)
P_1	-1.5	0.25	ψ_1	$\frac{\pi}{4}$	86
P_1	-1.5	0.25	ψ_2	$-\frac{\pi}{4}$	87
$*P_1$	-1.5	0.25	$*\psi_2$	$-\frac{\pi}{4}$	90
P_2	-1.5	0.00	ψ_1	$\frac{\pi}{4}$	79
P_2	-1.5	0.00	ψ_2	$-\frac{\pi}{4}$	77
P_3	-1.5	-0.25	ψ_1	$\frac{\pi}{4}$	76
P_3	-1.5	-0.25	ψ_2	$-\frac{\pi}{4}$	77

$CovReq$ was achieved before turning to enter the ideal orbit. Remarkably, the $P_1\psi_1$ path instead approached the target off-center. This was due to the decrease in sensor covariances along the diagonals between the East and North coordinate axes. The covariance trend was noticeable in Figure 4.1b. This effect was a result of the linearizing approximation method used to convert from the sensor frame to the KF's ENU coordinate system.

When optimized, the $P_1\psi_2$ trial displayed an extreme sensitivity to the initial control vector provided. Local minimums could be achieved by flying North or South of the obstruction NFZ, however the Southern path was selected by the operator due to its optimal t_1 (87 instead of 90 seconds). The Northern solution from the $*P_1*\psi_2$ trial approached the target at a similar angle to the $P_1\psi_1$ solution. The difference in orbit direction was possibly due to the dynamic NFZ having no effect on the slower solution.

4.8 Effects of Constant Wind

The effects of different wind conditions on the scenario studied in Section 4.5 were compared with all other conditions, constraints, and cost function weights held constant. Figures 4.13 and 4.14 display the different optimal paths and control vectors, respectively. The exact wind conditions and targeting times are tabulated in Table 4.6 as well as a qualitative description of the flight path with respect to the dynamic NFZ and wind direction.

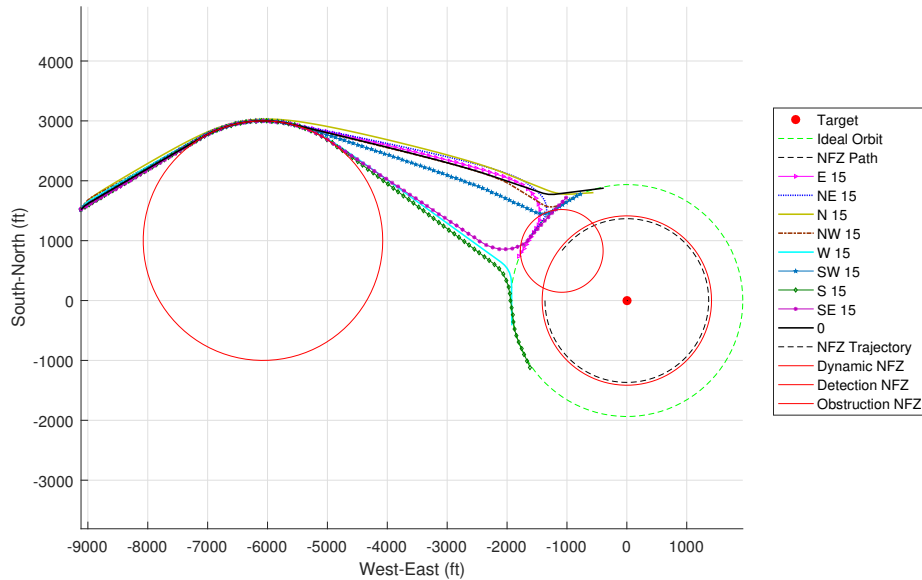


Figure 4.13. Comparison of the NFZs and trajectories for different wind conditions

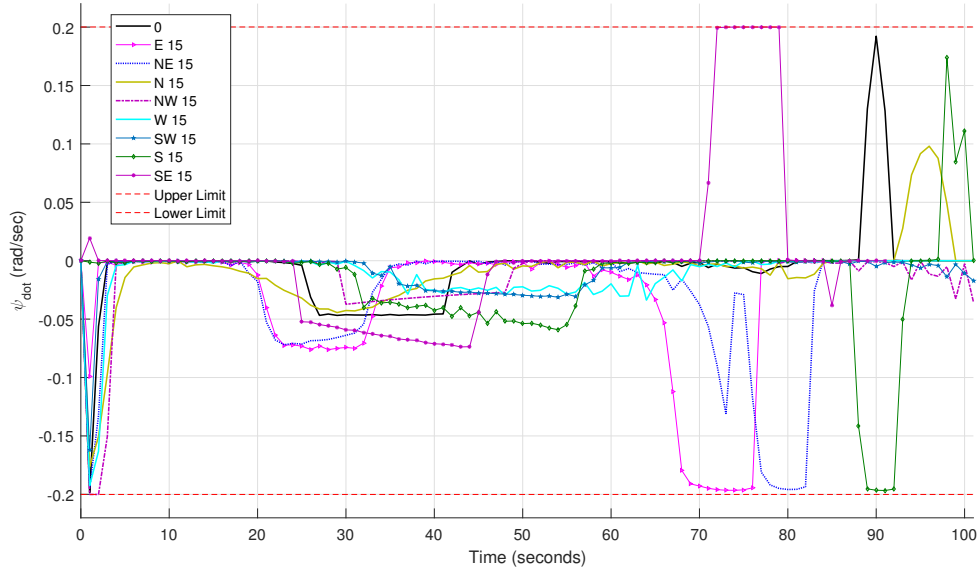


Figure 4.14. Comparison of the control vectors for different wind conditions

Several quantitative trends were apparent from the simulation results. As expected, targeting time was the smallest when the UAV initial position was upwind of the target. The targeting time of each case with constant wind was compared to the base value for targeting time from the no-wind scenario in column 3 of Table 4.6. For the 8 different 15 kt wind conditions, there was at worst a 36% increase in targeting time t_1 , and at best a 19% decrease in t_1 . Scenarios with the wind near-perpendicular to the path of the UAV had a negligible change in t_1 . Thus, an optimal, upwind launch will decrease the UAV endurance cost of target approach by approximately 20% compared to a no-wind or crosswind launch. Compared to a worst-case downwind launch, the endurance cost is decreased by almost 40%. For a 90 second flight segment, this was a minor improvement. However if the vehicle launch point was from 5-10 NM away from the target, endurance savings would be significant and increase the UAV's potential time-on-target.

Qualitatively, the optimal approach and entry to target orbit displayed several trends. The direction of orbit entry, clockwise (CW) or counterclockwise (CCW),

Table 4.6. Changes in Optimal Path and Targeting Time Due to Wind Conditions

Wind (kt)	t_1 (sec)	t_1 Change	Path Description
None	86	N/A	Approach off-center, hard turn towards CW orbit past dynamic NFZ
15 East	70	19% Decrease	Approach off-center downwind, hard turn into upwind CCW orbit to avoid dynamic NFZ
15 North East	74	14% Decrease	Approach off-center downwind, hard turn into upwind CCW orbit to avoid dynamic NFZ
15 North	88	2% Increase	Approach off-center downwind to enter crosswind CW orbit past dynamic NFZ with minimal control
15 North West	106	23% Increase	Approach off-center tacking upwind until t_1 , turn directly upwind toward target, turn into crosswind CW orbit (dynamic NFZ no effect)
15 West	115	36% Increase	Approach target tacking upwind, turn into crosswind CCW orbit (dynamic NFZ no effect)
15 South West	109	27% Increase	Approach off-center tacking upwind, hard turn into upwind CW orbit (dynamic NFZ no effect)
15 South	90	5% Increase	Approach target downwind, hard turn into downwind CCW orbit (dynamic NFZ no effect)
15 South East	76	12% Decrease	Approach target (downwind), hard turn into crosswind CW orbit (dynamic NFZ no effect)

was generally arbitrary unless the dynamic NFZ forced immediate deviation from the ideal orbit in one direction. The optimal determination of orbit direction is studied in Section 4.9.

4.9 Steady-State Target Observation

Two different sets of initial conditions were modeled to compare the “steady-state” orbit of the target in co-rotation or counter-rotation with the dynamic NFZ. Depending on the ideal orbit radius (which was based on a fixed ideal slant range), UAV speed, UAV altitude, and dynamic NFZ speed, several scenarios of interest were possible. A flight altitude of 1,000 ft AGL was selected to emphasize the influence of the dynamic NFZ. At lower altitudes, the dynamic NFZ barely extends outside of the detection NFZ and has a minimal effect on maintaining the ideal orbit.

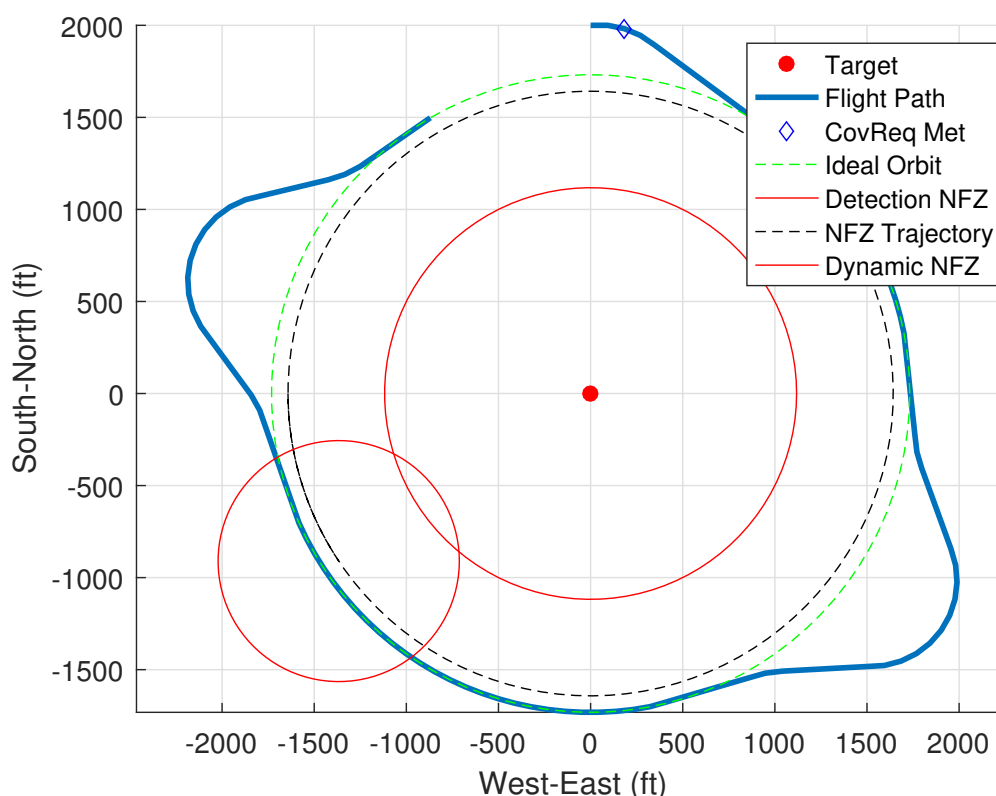


Figure 4.15. Orbit of the target at 1,000 ft AGL in counter-rotation with a dynamic NFZ

In Figure 4.15, the UAV orbited in a clockwise direction, opposite of the dynamic NFZ. Because the dynamic NFZ and UAV have similar speeds about the target at

this ideal orbit radius (V_{NFZ} is slightly faster than V_{UAV}), the UAV must deviate from the ideal orbit approximately twice every revolution. Regardless, the targeting covariance remains below $CovReq$ throughout each deviation from the ideal orbit. A larger steady-state orbit could be selected by the operator to remain outside of the NFZ's outer radius without deviation and $CovReq$ would be maintained.

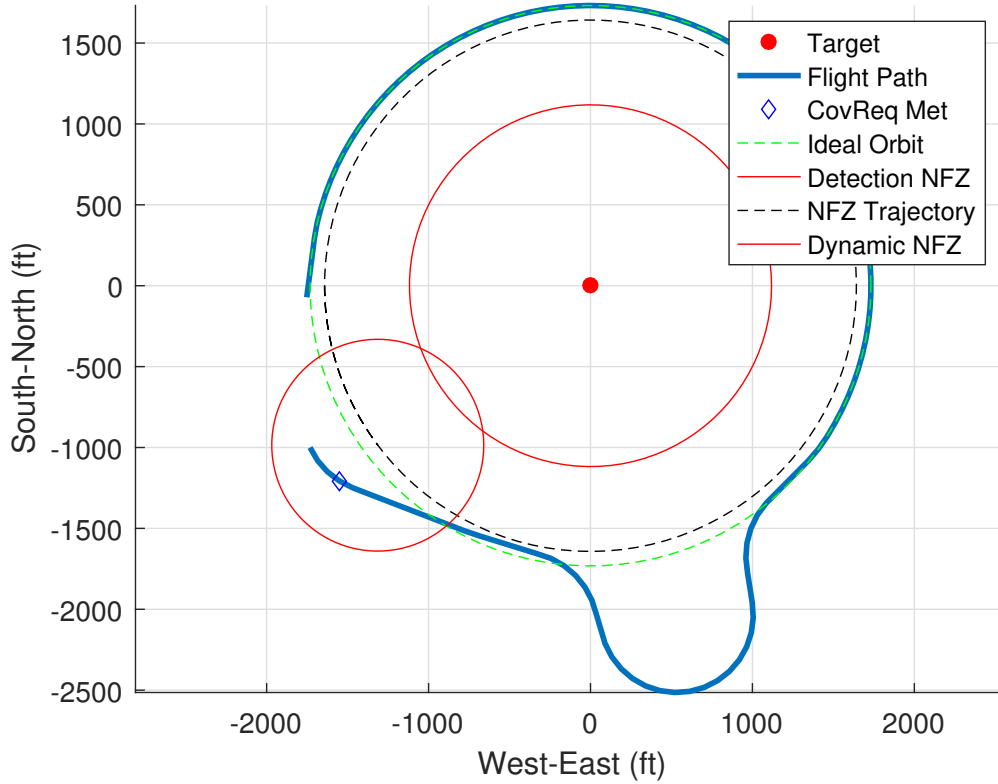


Figure 4.16. Orbit of the target at 1,000 ft AGL in co-rotation with a dynamic NFZ

In Figure 4.16, the UAV orbited in a counter-clockwise direction: the same direction as the dynamic NFZ. Because V_{NFZ} is slightly faster than V_{UAV} , the UAV must deviate from the ideal orbit only once in many revolutions (depending on the ideal orbit radius). As the targeting covariance remained below $CovReq$ throughout the deviation and ideal orbit, targeting fidelity was maintained. The deviation from

the ideal orbit was larger than in the counter-clockwise case, but occurred on a much lower frequency.

The choice of co-rotation or counter-rotation was left to the operator's preference as both orbits maintain targeting fidelity. Regardless, the orbit could be deconflicted simply by increasing the specified ideal orbit radius.

4.9.1 Steady-State Observation with an Orbit Obstruction

A unique situation was contrived to determine the ideal flight path when a traditional orbit was impossible due to a large obstruction such dangerous airspace. Due

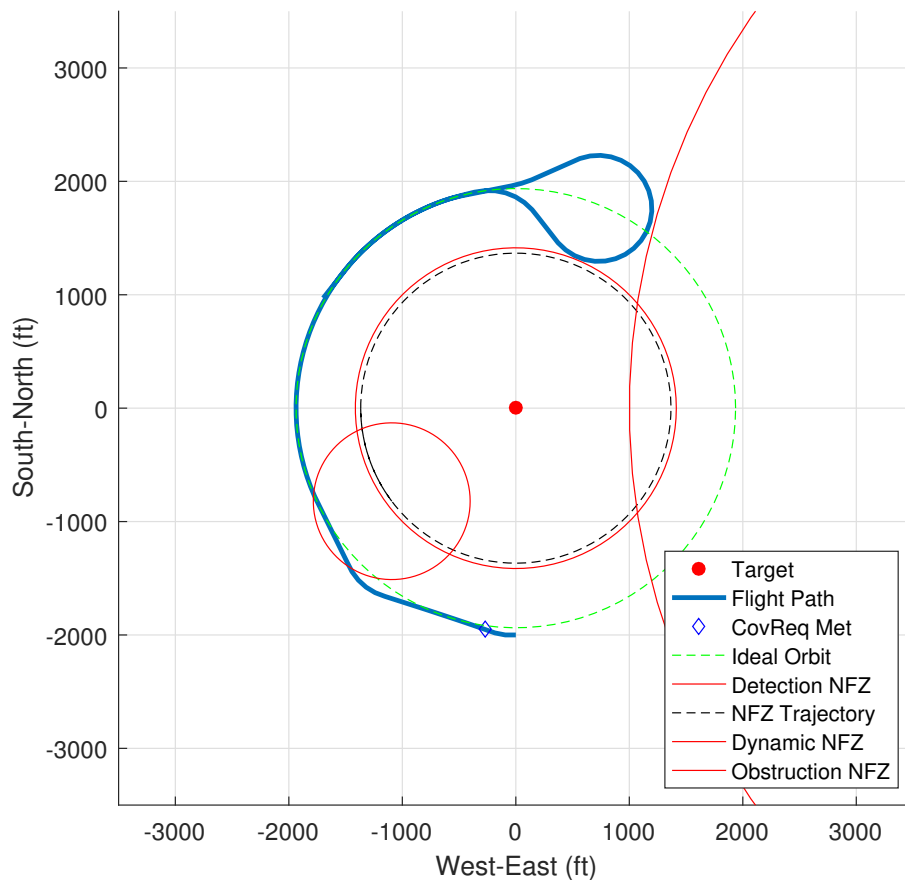


Figure 4.17. Attempted orbit of the target at 500 ft AGL with an obstruction NFZ

to extensive processing time, this scenario was only simulated with $t_2 = 120$ seconds. A simulation with a much higher t_2 would offer an exact and complete steady-state solution. However, this two minute long trajectory is sufficient to qualitatively determine the shape of the optimal steady-state trajectory. With the full cost function (which weights covariance, control effort, and distance from the ideal orbit radius), the optimal trajectory's turnaround was centered about the ideal orbit radius with a maximum $\dot{\psi}$ (shown in Figure 4.18) only for the u-turn segment.

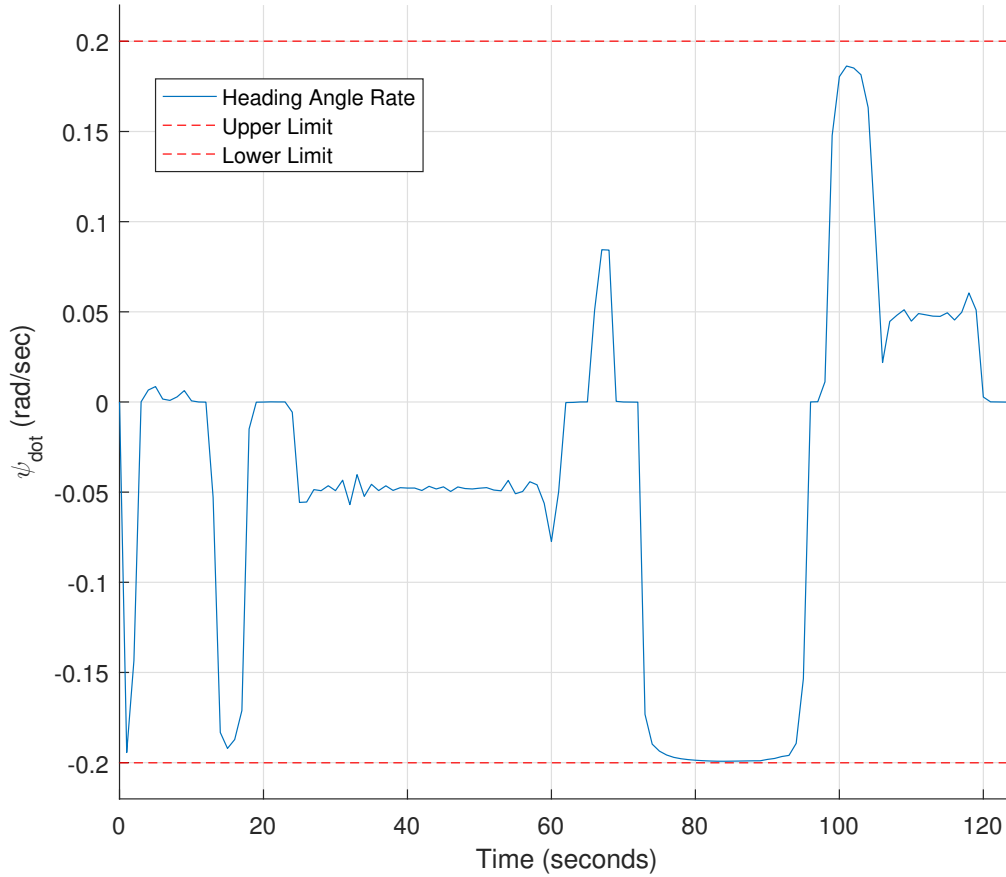


Figure 4.18. The optimal control for the obstructed orbit

By a logical extension of this simulation, the optimal path will follow the ideal orbit (deviating to avoid the dynamic NFZ when necessary) with a turn around

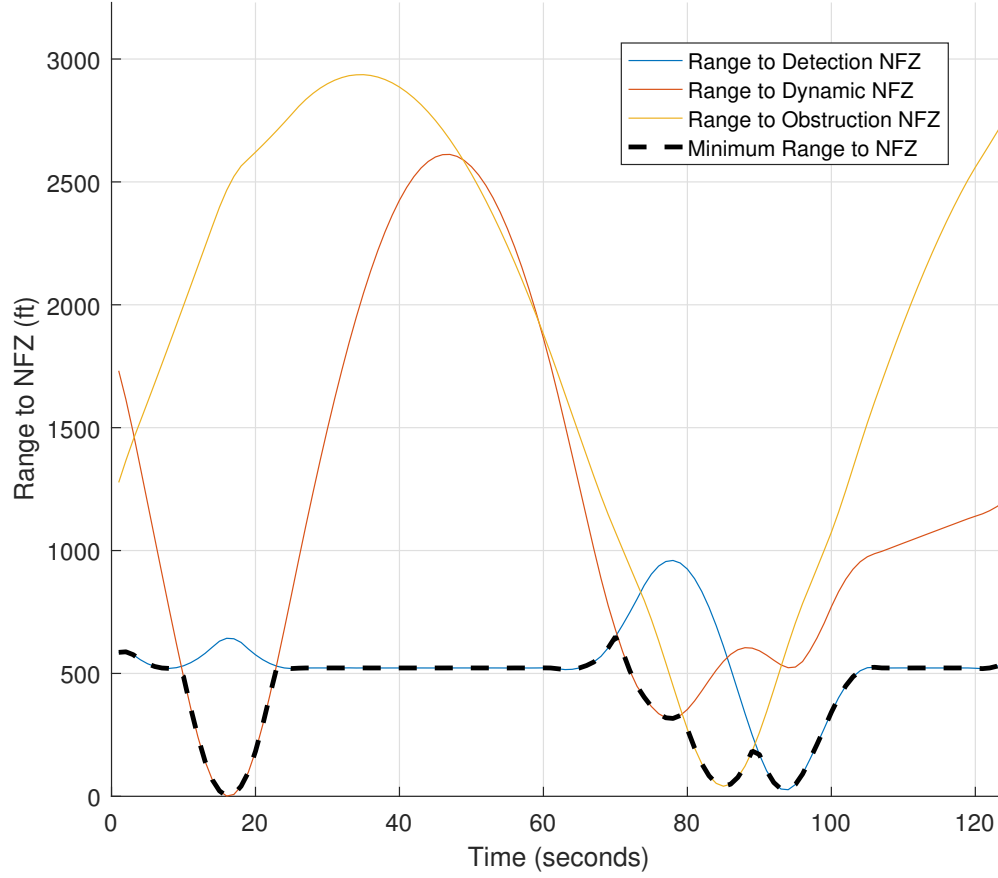


Figure 4.19. Distances to the NFZs with respect to time for the obstructed orbit

(in the shape shown in Figure 4.17) just before the obstruction NFZ. A special case might occur to avoid both the dynamic and obstruction NFZs with a single maneuver, however this was not modeled due to extensive processing time.

As shown in Figure 4.20, this maneuver maintains a target covariance well below $CovReq$. Recall that the sensor was assumed to have no limitations on look angle or turn rate. So long as the sensor on the actual UAV is capable of taking LOB measurements from all involved aircraft orientations with respect to the target, no targeting fidelity will be lost in the maneuver.

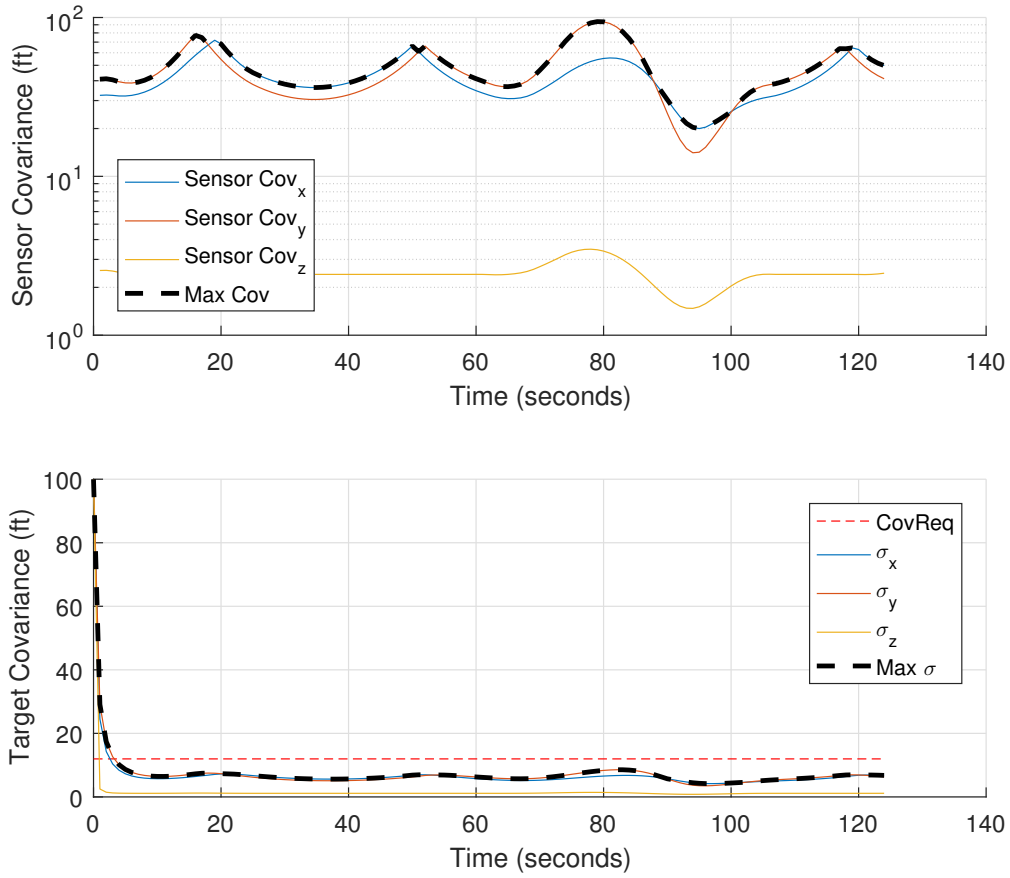


Figure 4.20. Sensor and target covariance for the obstructed orbit

4.10 Algorithm Performance

The scenario shown in Section 4.5 was optimized several times with different bootstrapper settings to compare algorithm performance. The performance of the different bootstrapper methods and settings was summarized in Table 4.7. All trials were given an initial value for t_f of 60 seconds. This low initial value of t_f guaranteed many iterations and extensions of the bootstrapper. The specified t_2 was for all trials was 30 seconds. The settings used to generate the paths shown throughout Chapter IV recorded in the first row of the table, denoted by the asterisk. Processing time was dependent on computer hardware, thus all compared trials were run on a Lenovo

X1 Carbon laptop with Intel®Core i7-6600U CPU @ 2.60GHz, 16 GB RAM, and 1 TB SSD.

Table 4.7. Performance of the Bootstrapping Algorithm Under Different Settings

Rule	Ratio/StepSize	StepTol	fmincon Alg	Runtime (mm:ss)	Comments
*Ratio	1.25	1e-6	‘interior-point’	11:55	
Ratio	1.25	1e-6	‘sqp’	10:43	Solution Unfeasible
Ratio	1.50	1e-6	‘interior-point’	15:59	
Ratio	1.10	1e-6	‘interior-point’	14:25	
Step	1	1e-6	‘interior-point’	44:24	Smoothest Solution
Step	5	1e-6	‘interior-point’	19:36	
Step	10	1e-6	‘interior-point’	17:29	

The default method with the extend ratio of 1.25 and ‘interior-point’ algorithm ran the fastest and most reliably of the options tested. The relevant bootstrapper decisions and actions of the optimization script run on the default settings are summarized below.

```

INITIALIZE with tf=60 at 00:00
    Optimization completed at 00:50
    CovReq not attained
EXTEND to tf=75 (by ratio)
    Optimization completed at 02:42
    CovReq not attained
EXTEND to tf=94 (by ratio)
    Optimization completed at 05:44
    CovReq attained after t1 but before tf
EXTEND to tf=115 (by t2 buffer)
    Optimization completed at 09:59
    CovReq attained before t1
TRIM to tf=114 (based on CovReq and t1)
    Optimization completed at 11:25
    CovReq attained at t1
    CovReq not attainable before t1
QUERY user for input
    Do not decrease StepTol
    Do not optimize with different algorithm
    User interaction completed at 11:50
RETURN Optimal Path
    Generate plots
    Save optimal vector
END script at 11:55

```

In general, the algorithm converged on an optimal solution the fastest when given an initial guess for t_f (the total length of the state vector in seconds) close to the final solution. Each iteration in which the bootstrapper extended the state vector would re-explore the same trajectories as before with no memory of its previous iterations due to restarting of `fmincon` with each bootstrapped iteration. The extended trajectory would usually maintain its shape for the previous portion (which was faster than starting each iteration over with a zero control vector). Each method did converge

to the same solution. The only differences in the control vector (due to the *StepTol* of 10^{-6}) were small amounts of noise. The paths and values for t_1 were essentially identical.

One potential source of the long run times shown in Table 4.7 was the use of an absolute value function within the calculation of the cost function and inequality constraints as shown in Equations 3.46 and 3.47. The absolute value function can, in some cases, cause difficulty in the numeric calculations of gradients and Hessians needed to converge on the optimal solution efficiently.

The bootstrapping method with the constant time step showed a strong sensitivity to the initial guess, especially with higher initial values of t_f . If the initial trajectory passed clockwise around a NFZ, `fmincon` would rarely find a solution which passed counterclockwise around the NFZ due to a loss of feasibility in intermediate iterations. Accordingly, some of the trials with multiple possible trajectories around the static “obstruction NFZ” required the operator to input a modified initial state vector to find other locally optimal solutions.

4.11 Summary

Through independent variation of the scenario and algorithm parameters, the effects of different cost functions, wind conditions, flight altitudes, and orbit directions were studied. By inspection of the optimal solutions, different trajectory characteristics and trends were realized which could later be compiled into a real-time, near-optimal path planner ideal for onboard processing. As well, the development and use of the bootstrapping method to minimize t_1 through the iterative minimization using an alternate cost function revealed several benefits, shortcomings, and effects. Due to the proposed sensor specifications of the simulated UAV, variation of the sensor

accuracy was not studied. Thus, the effects of sensors with different uncertainties and covariances were not studied.

Overall, the UAV trajectories generated for the various trials not only proved the functionality of the optimal path planning algorithm, but led to several significant conclusions for both TOBS operation and path planning algorithm development.

V. Conclusions and Recommendations

This thesis developed a path planning algorithm to determine the minimum targeting time solution for a Tactical Off-Board Sensing (TOBS) UAV. To minimize the time required to drive a geolocated target covariance below a specified threshold, a bootstrapped method was created to approach a minimum time solution to achieve a targeting covariance requirement by iteratively minimizing final covariance with a fixed time step discretization.

By breaking down the optimization algorithm into sensible components and functions, tools for both path planning and operational target estimation were developed. The approximation method for linearizing Line-of-Bearing measurements into the ENU coordinate frame and the discrete KF operated in unison to estimate a target's position and track the covariance both in simulation or real-time operation.

The control vector discretization allowed for the deconfliction of the UAV path with a number of known static and dynamic NFZs, and the use of bounds on the control vector enforced the model dynamics. As well, a tuneable cost function allowed the operator to prioritize between minimizing targeting time, control effort, deviation from an ideal orbit radius, and target covariance at every time step.

By implementing relevant values for simulation settings and contriving scenarios to emulate real-world situations, useful estimates on mission performance and capabilities were found and the characteristics of optimal trajectories were assessed. Information specific to the TOBS mission can serve to improve and guide decisions during UAV testing and development and when developing the Tactics, Techniques, and Procedures (TTPs) required for operation. Beyond the scope of TOBS, the proposed methods of optimization and other tools developed may serve as a guide for future development in UAV path planning.

5.1 Conclusions from Simulated Results

Based on the simulated results of the real-world scenarios proposed in Sections 4.5, 4.8, and 4.9, several estimates and predictions on targeting performance were found. From a proposed launch 1.5 nautical miles away from the target, targeting was attainable in under 90 seconds for all simulated conditions (except for a headwind, when targeting took up to 2 minutes to complete). At 500 feet AGL, targeting was achieved at ground distances to the target of approximately 0.38 NM. At 1250 ft AGL, targeting was achievable at a approximate ground distance of 0.54 NM and time of 76 seconds. The results for each of the simulations run were presented in Chapter IV. The significance of these results, however, is that geolocation of the target does not require entering and maintaining an orbit for long periods of time given the problem assumptions. Sufficient targeting certainty could be found during the approach to a target, enabling traditional orbits to be foregone if constant surveillance were not required. Additionally, once *CovReq* was attained, the targeting certainty was proven to be maintained throughout the maneuvers necessary to avoid NFZs or switch orbit direction assuming no visual obstructions to LOB measurements.

When comparing launch conditions in Sections 4.8 and 4.7, path improvements due to optimal wind conditions and initial positions were quantified. In review, for trials with an obstruction NFZ, Unmanned Aerial Vehicle (UAV) launch upwind of the target provides a 19% decrease in targeting time from the base case, or a 40% decrease in targeting time from the worst case launch scenario. Similarly, selecting launch points unaffected by the 2000 ft radius obstruction NFZ yielded a 10% decrease in targeting time. While these decreases in targeting time were simply comparisons of contrived scenarios, they provide baseline estimates which could be later used to determine the optimal mid-air launch of a UAV from a secondary air vehicle or to determine the benefits of different ground launch location.

An important consideration is that these results were based on the assumption of “perfect” measurements, with sensor noise accounted for as a covariance. This assumption was used to ensure `fmincon` convergence. Thus, all calculated final times were slightly faster than is realistically possible given actual sensor noise. Any real Line-of-Bearing (LOB) measurement with noise would shift the target estimate slightly and increase the target covariance more than was simulated.

5.2 Identified Traits of Optimal UAV Trajectories

Based on the simulated results from variations of trajectory parameters (orbit direction, altitude, wind direction, initial conditions), the generation of a near-optimal trajectory in a variety of conditions could be performed using several trends and geometry. Due to the 3-dimensional aspect of the LOB measurements, the “bearing-only” contained implicit ranging information. This caused the trajectories to closely resemble Dubin’s paths. As a result, the necessary data and process needed to generate near-optimal trajectories is simpler.

When determining the optimal launch point, wind direction and ground distance to the target were the main influences on minimizing final time. Under the assumption that all TOBS launches must occur at some minimum distance from the target, the ideal launch location was directly upwind of the target (according to Tables 4.6), as close to the target as possible. If a large obstruction No-Fly-Zone (NFZ) were present between the upwind launch point and target, the optimal launch point would be shifted to decrease the deviation from the straight line path to the target while maintaining a downwind flight path. To determine the optimal offset of the launch point due to a NFZ, the time of flight to the target (represented by a straight segment, curved segment, and straight segment) would be minimized using calculus.

A simple and efficient solution for this problem would be a useful contribution for future research.

When determining the optimal altitude for a TOBS mission, both the minimization of targeting time and requirement for target identification (which was not studied in this research) must be accounted for. Based on the results from Table 4.4, a flight altitude of 1250 ft AGL was preferable. If the ceiling on flight altitude due to identification requirements was lower than 1250 ft, flight at the highest allowed altitude would be optimal (based on the cost function utilized in this research).

When determining the optimal direction of orbit for steady-state observation, the ideal slant range and dynamic NFZ trajectory, speed, and size influence the decision. If the dynamic NFZ remained entirely of the detection NFZ, then orbit direction has no effect on optimality and would be entered in the direction of minimum control effort or operators preference. In accordance with the trials shown in Figures 4.15 and 4.16, if the dynamic NFZ extends significantly into the ideal orbit's path and $V_{NFZ} \approx V_{UAV}$, a co-rotating orbit would result in the fewer deviations from the steady-state orbit. However, a full study at the operationally selected altitude would provide a better basis for the decision.

Given the data of the simulated trials, the development of a near-optimal, real-time path planning algorithm for onboard use is a logical next step for future work.

5.3 Conclusions on the Optimization Algorithm

During the research process, there were two primary obstacles to success: the bootstrapping algorithm and the method of linear approximation of sensor covariances for estimation. The first problem was a result of desiring the ability to model airframes or sensors with a fixed time step discretization. Because of the issues inherent to the minimum final time problem with fixed time step discretization, the development of

the bootstrapping method was, at the time, a necessary enterprise. As proven by the many optimization trials run, the iterative method of minimizing final covariance and trimming or extending the state vector to approach a minimum final time solution was successful. However, the algorithm resulted in a redundant (and sometimes inefficient) exploration of similar solutions with different state vector lengths. Values for the initial t_f guess close to or above the optimal value led to a shorter runtime. The actual bootstrapping (feeding of the previous iteration’s optimal solution to the input of the next iteration) unquestionably sped up the iterative finding of the optimal solution, but leaves room for future improvement on efficiency. Several methods for this improvement were recommended below in Section 5.4.

The second largest obstacle of this research was the development of a complete and efficient estimation method, with measurements in the moving sensor frame with the estimate desired in the ENU coordinate frame. Due to the desire for a system capable of estimating the target state and its covariance in both simulation and real-time operation, the iterative KF was selected. The issue of converting from the sensor frame to any flat-Earth-fixed coordinate system resulted in an approximation method. The assumptions required for the approximation method led the author to believe that a more comprehensive and appropriate method exists. The approximation method was settled on because it was only one of many components of the research, and does satisfactorily serve its purpose. Due to the use of the approximation method, no comparison of LOB measurements with and without range measurements was studied. Recommendations for future research in this area are recorded in Section 5.4.

Another unique aspect of this optimization method was the selection of the state and control vector. In other iterations of the algorithm’s development, a state vector was utilized which tracked the UAV’s x , y , and ψ coordinates for each time step, with

ψ being the “control”. The state vector of length $3N$ often led to state vectors several hundred elements long in extended scenarios which optimized slowly. The method in this research (the state vector containing only $\dot{\psi}$) shortened the state vector, but led to sensitivity issues in some situations. If a loop or circle formed in the trajectory, a obviously sub-optimal local minimum could be found by *fmincon*. The creation of loops in *fmincon*’s iterations was more common in the $\dot{\psi}$ vectorization than the x , y , ψ vectorization. Essentially, the state vector was more sensitive to a change in heading rate early in the path than it was to a similar change in x , y , and ψ due to the enforcement of path constraints. This comparison could merit closer study in future work.

5.4 Recommendations for Future Work

The bootstrapping method of iteratively approaching the minimum final time solution while utilizing a fixed time step offers several avenues for future work. Improved “smart” methods for extending the state vector by a ratio scaled by the current solution’s cost might significantly decrease the necessary iterations and overall runtime to optimize a trajectory. Alternatively, a cost function could be developed to avoid the bootstrapping by dynamically sliding the indices (such as N_{t1}) which mark the boundaries between phases. A state vector longer than any expected solution could be efficiently optimized, with any extra “dummy states” after tf being ignored or weighted as zero. Alternatively, the use of a pseudospectral method would decrease the optimization runtime. A pseudospectral approach may run fast enough for use as an onboard optimal trajectory planner or in extensive simulation. Finally, a heuristic solution could be developed to plan near-optimal trajectories based on simulated results.

The next area for future research in this area is the development of a more com-

prehensive sensor model and estimation method. More importantly, the development of an Extended KF or iterative least-squares estimator may remove the need for the linearizing approximation used in this research. Along with eliminating the imperfect approximation, a more complete sensor model would transfer to real-world operation better and yield more relevant results. Ideally, the location of a new estimator would be reported in ENU coordinates, and the target covariances would be orthogonal with each other, but not necessarily collinear with the ENU axes. With an improved estimator, a full comparison of the effects of LOB measurements with and without range measurements could be studied.

Especially due to the existence of dynamic NFZs, the expression and enforcement of NFZs merits further study. An alternative cost function could place a weight on time spent in or near NFZs, so as to minimize “risk” during a flight while also minimizing final time, deviation from the ideal orbit, or other priorities. Rather than enforcing the NFZs as inequality constraints, they could be penalized heavily in the cost function. Additionally, allowing for variable altitude would provide a larger solution space for the optimization method. NFZs could be modeled as 2D slices of 3D shapes or as “Superquadrics” (3 dimensional parametric functions) [22].

Finally, flight testing of the optimal trajectories found in this research or of near-optimal trajectories based on the findings reported in Section 5.2 would be extremely valuable. Real, noisy measurements would test the abilities of the KF and magnify any incompatibilities between the optimal simulated trajectories and the actual TOBS aircraft.

5.5 Closing Remarks

After exploring many different paths to develop and test a complete optimization algorithm, a solution has been found. It may not be the most efficient or advanced

method of dynamic optimization, but in this field there may in fact be no such thing. The lessons learned and methods practiced have created a product with real application. More importantly, the many attempts which failed have created a stronger and more experienced scholar. This research, work, and constant learning will be continued at the Air Force Research Laboratory.

Appendix A. Cost Function MATLAB Code

The function `CostFun_MinT1Cov_Orb` takes the control vector as an input and returns the resulting scalar cost. K_{f_t1} is the weight on maximum component of covariance at $t1$. $K_{f_t2_cov}$ is the weight on the maximum components of covariance at each time step after $t1$. $K_{f_t2_orb}$ is the weight on distance from the ideal orbit for every time step after $t1$. Finally, K_{f_ctrl} is the weight on control for every time step.

```

1 function f = CostFun_MinT1Cov_Ctrl_Orb(X)
2     % CostFun_MinT1Cov minimizes:
3     %   Covariance at t1
4     %   Control effort for all steps
5     %   Covariance at each step after t2
6     %   Distance from ideal orbit for each step after t2
7
8     % Declare necessary global variables
9     global N N_t2 IdealOrbRad X_t N_t1
10    psi_d_v = X;
11    [ x_v, y_v, ~ ] = getPathFromPsi_d( psi_d_v );
12
13    % Get histories of kalman filter
14    [ P_k_k_hist, ~, ~ ] = GetKalmanHistory(x_v,y_v);
15
16    % Account for covariance at t1 (PRIMARY FOCUS)
17    f_t1 = max(P_k_k_hist(N_t1,:)); % Max covariance at N_t1
18
19    % Account for control effort
20    f_ctrl = sum(abs(psi_d_v));
21
22    % Account for covariance at each step after t1
23    f_t2_cov = sum(max(P_k_k_hist((N-N_t2):N,:), [], 2));
24
25    % Account for distance from ideal orbit radius after t1
26    f_t2_orb = sum(abs(IdealOrbRad - ...
27        sqrt((x_v(N_t1:end)-X_t(1)).^2 + (y_v(N_t1:end) - X_t(2)).^2)));
28
29    % Assign weights to each component (Normal means max value ...
30        unweighted)
31    K_f_t1 = 3000;           % Normal:    11      Scaled:    33000
32    K_f_t2_cov = 10;        % Normal:    800     Scaled:    8000
33    K_f_t2_orb = 2;         % Normal:   13000   Scaled:   26000
34    K_f_ctrl = 1000;        % Normal:    7      Scaled:    7000
35
36    f = K_f_t1*f_t1 + ...
37        K_f_t2_cov*f_t2_cov + ...
38        K_f_t2_orb*f_t2_orb + ...
39        K_f_ctrl*f_ctrl;
40 end

```

Appendix B. No-Fly-Zone Constraints MATLAB Code

The function `Constraints_NFZ` takes the control vector as an input and returns a vector containing the distances from the UAV to each NFZ at every time step.

```
1 function [g, h] = Constraints_NFZ(X)
2
3     %%% Declare necessary global variables
4     global N Num_NFZs;
5     global R_NFZ_1 X_NFZ_1 Y_NFZ_1;
6     global R_NFZ_2 X_NFZ_2 Y_NFZ_2;
7     global R_NFZ_3 X_NFZ_3 Y_NFZ_3;
8
9     %%% Pull out states from current vector
10    psi_d_v = X(1:N); % Heading Angle Rates of UAV
11
12    % Given psi_d_v, calculate the path of the vehicle
13    [ x_v, y_v, ~ ] = getPathFromPsi_d( psi_d_v );
14
15    %%% Enforce NFZ with inequality constraints for each time step
16    % Zero NFZs
17    if Num_NFZs<=0
18        g = []; % Inequality Constraints
19    end
20
21    % One NFZ
22    if Num_NFZs>0
23        g = zeros(Num_NFZs*N,1); % Pre-allocate vector for efficiency
24        for i=1:N
25            g(i) = R_NFZ_1(i) - ...
26                sqrt((x_v(i) - X_NFZ_1(i))^2 + (y_v(i) - Y_NFZ_1(i))^2);
27        end
28    end
29
30    % Two NFZs
31    if Num_NFZs>1
32        for i=1:N
33            g(N+i) = R_NFZ_2(i) - ...
34                sqrt((x_v(i) - X_NFZ_2(i))^2 + (y_v(i) - Y_NFZ_2(i))^2);
35        end
36    end
37
38    % Three NFZs
39    if Num_NFZs>2
40        for i=1:N
41            g(2*N+i) = R_NFZ_3(i) - ...
42                sqrt((x_v(i) - X_NFZ_3(i))^2 + (y_v(i) - Y_NFZ_3(i))^2);
43        end
44    end
45
```

```

46     % Catch Errors
47     if Num_NFZs>3
48         error('Number of NFZs not implemented yet');
49     end
50
51     %%% Help with feasibility
52     g = g - (1e-6);
53
54     %%% No equality constraints needed
55     % Path constraints enforced by UB and LB on control
56     h = [];
57 end

```

Appendix C. Line-of-Bearing Measurements and Approximation MATLAB Code

The function `GetLOB` generates a LOB measurement from the UAV to the target in spherical coordinates in the sensor frame.

```
1 function [ psi, theta, r ] = GetLOB( X_sens, X_targ, units)
2     %GetLOB Given UAV and target cartesian coords, returns LOB (with or
3     %without ranging measurement to target. This simulates a real LOB
4     %measurement from the UAV without noise.
5     %
6     %   INPUTS:
7     %       X_sens  Exact position of sensor in [X;Y;Z]
8     %       X_targ  Exact position of target in [X;Y;Z]
9     %       units   Specify 'deg' or 'rad', default is radians
10    %
11    %   OUTPUTS:
12    %       psi      Azimuth angle (angle from East, 90 deg is north)
13    %       theta    Elevation angle (angle from vertical, 0 is down)
14    %       r        Slant range in same distance units as X_sens & ...
15    %       X_targ
16    %   NOTES:
17    %       If only one output is requested, [psi, theta, r] is ...
18    %       returned as
19    %       an array instead of as three separate values.
20    % Check if input dimensions match
21    if length(X_sens) ~= length(X_targ)
22        error('Dimensions of X_sens and X_targ do not match');
23    end
24
25    % Convert to '3D' if inputs are just XY
26    if length(X_sens)==2
27        X_sens = [X_sens(1) X_sens(2) 0];
28        X_targ = [X_targ(1) X_targ(2) 0];
29    end
30
31    % Create cartesian vector from sensor to target, convert to ...
32    % spherical
33    [psi,theta,r] = cart2sph(X_targ(1)-X_sens(1),...
34        X_targ(2)-X_sens(2),...
35        X_targ(3)-X_sens(3));
36
37    % Convert to degrees if requested.
38    if nargin==3
39        if strcmp(units,'deg')
40            psi = rad2deg(psi);
41            theta = rad2deg(theta);
42        elseif strcmp(units,'rad')
```

```

42         % Do nothing
43     else
44         warning('Units not recognized, default to radians')
45         % Do nothing
46     end
47 end
48
49 % Output results as an array if only one output arg requested
50 if nargout==1
51     psi = [psi theta r];
52 end
53 end

```

Once the LOB measurements are generated, the function `LOB2xyzWithCov` converts the measurements and covariances from the sensor frame to the ENU frame.

```

1 function [ X_meas, Cov_xyz, range, Cov_ptr] = LOB2xyzWithCov( ...
    X_sens, ...
2     X_targ_est, LOB, Sv)
3     %LOB2xyzWithCov Given sensor position, target position estimate, ...
    LOB,
4     %and sensor covariances, returns the "measured" target location with
5     %XYZ covariances. Works with or without LOB range measurement.
6     %
7     %   INPUTS:
8     %       X_sens      Exact position of sensor in [X;Y;Z]
9     %       X_targ_est  Current target position estimate in [X;Y;Z]
10    %       LOB          [psi, theta, (r)] from sensor to target in ...
    RADIANS
11    %       Sv          Sensor Covariances diag([psi, theta, (r)])
12    %
13    %   OUTPUTS:
14    %       X_meas       [X;Y;Z] of measured target location
15    %       Cov_xyz      XYZ covariance of measurement
16    %       r            "Measured" range to target (if not provided)
17    %       Cov_ptr      Covariance of measurement due to psi, theta, r
18
19    % Two methods are required: with and without range measurements
20    if length(LOB)==3 % If range given, this is easier.
21        range = LOB(3); % This is already known
22        % Calculate XYZ location of measured target
23        [deltaX,deltaY,deltaZ] = sph2cart(LOB(1),LOB(2),range);
24        X_meas = X_sens + [deltaX; deltaY; deltaZ];
25
26        % Convert Sensor covariances from angles to distances using ...
        trigonometry.
27        sigma_psi = 2*range*tan(Sv(1,1)/2);
28        sigma_theta = 2*range*tan(Sv(2,2)/2);
29        sigma_r = Sv(3,3); % No conversion necessary
30

```

```

31 else % If the range is not given, this is more difficult.
32     % Calculate a point along the sensor bore sight
33     [X_sens_x, X_sens_y, X_sens_z] = sph2cart(LOB(1),LOB(2),1);
34     X_sens_pt = X_sens + [X_sens_x; X_sens_y; X_sens_z];
35
36     % Calculate the range from the sensor to the point closest ...
37     % to the
38     % target position estimate.
39     % ...
40     % {http://mathworld.wolfram.com/Point-LineDistance3-Dimensional.html}
41     % range = -dot((X_sens-X_targ_est),...
42     % (X_sens_pt-X_sens))/norm(X_sens_pt-X_sens)^2;
43     range = -((X_sens-X_targ_est)'*(X_sens_pt-X_sens))/...
44     % norm(X_sens_pt-X_sens)^2;
45     % Convert known X_sens, psi, theta, and range to XYZ measurement
46     [deltaX,deltaY,deltaZ] = sph2cart(LOB(1),LOB(2),range);
47     X_meas = X_sens + [deltaX; deltaY; deltaZ];
48
49     % Convert Sensor covariances from angles to distances using ...
50     % trigonometry.
51     sigma_psi = 2*range*tan(Sv(1,1)/2);
52     sigma_theta = 2*range*tan(Sv(2,2)/2);
53
54     % Choose arbitrarily large sigma_r
55     sigma_r = inf; % Could be a finite approximate
56 end
57
58 % Create vectors in XYZ frame of psi, theta, r directions
59 [sigma_r_x, sigma_r_y, sigma_r_z] = sph2cart(LOB(1),LOB(2),1);
60 sigma_r_vec = [sigma_r_x; sigma_r_y; sigma_r_z];
61 sigma_psi_vec = cross(sigma_r_vec,[0;0;1]); % Cross with Z axis
62 sigma_theta_vec = cross(sigma_psi_vec, sigma_r_vec);
63
64 % Normalize to create unit vectors
65 sigma_psi_vec = sigma_psi_vec/norm(sigma_psi_vec);
66 sigma_theta_vec = sigma_theta_vec/norm(sigma_theta_vec);
67
68 % Multiply by weights and combine into "covariance" matrix as ...
69 % projected
70 % onto a perpendicular plane to the sensor at the target.
71 Cov_ptr = [sigma_psi_vec*sigma_psi, sigma_theta_vec*sigma_theta,...
72 -sigma_r_vec*sigma_r];
73
74 % NOTE: Now I need to extend these values onto the X, Y, and Z axes.
75 % This is a geometric calculation, not a vector projection. I should
76 % get 2-3 projections onto each axis. I can pick the smallest
77 % projections onto each axis as they should represent the most ...
78 % certain
79 % I can be within the XYZ coordinate system.
80
81 % Extensions of sigma_psi vector (which is parallel to ground)

```

```

78     sigma_x1 = sigma_psi/sin(LOB(1)); % Min value when psi==90, 270 deg
79     sigma_y1 = sigma_psi/cos(LOB(1)); % Min value when psi==0, 180 deg
80     sigma_z1 = inf; % In other words, this tells me nothing useful
81
82     % Extensions of sigma_theta vector
83     sigma_x2 = (sigma_theta/sin(LOB(2)))/cos(LOB(1));
84     sigma_y2 = (sigma_theta/sin(LOB(2)))/sin(LOB(1));
85     sigma_z2 = sigma_theta/cos(LOB(2)); % Min value when theta==0 deg
86
87     % Extensions of sigma_r vector
88     sigma_z3 = sigma_r/sin(LOB(2)); % Min value when theta==90 deg
89     sigma_x3 = (sigma_r/cos(LOB(2)))/cos(LOB(1));
90     sigma_y3 = (sigma_r/cos(LOB(2)))/sin(LOB(1));
91
92     % NOTE: Signs do not matter for the sigma values, only magnitude
93
94     % Determine which ones offer the lowest covariance.
95     sigma_x = min(abs([sigma_x1 sigma_x2 sigma_x3]));
96     sigma_y = min(abs([sigma_y1 sigma_y2 sigma_y3]));
97     sigma_z = min(abs([sigma_z1 sigma_z2 sigma_z3]));
98
99     % Present final result as matrix
100    Cov_xyz = diag([sigma_x, sigma_y, sigma_z]);
101
102    % Square the covariances
103    Cov_ptr = Cov_ptr*Cov_ptr;
104    Cov_xyz = Cov_xyz*Cov_xyz;
105 end

```

Appendix D. Discrete Kalman Filter MATLAB Code

The function `StepDKF` steps the discrete kalman filter used to estimate the target position and covariance forward one time step. When `StepDKF` is called for each time step, a full history of the target's location and covariance (as measured by the UAV) can be generated.

```
1 function [ X_hat_k1_k1, P_k1_k1 ] = StepDKF( Q_k, R_k1, P_k_k, ...
2     X_hat_k_k, Z_k1 )
3     %StepDKF Steps a Discrete Kalman Filter forward one time step. ...
4     %This is
5     %designed with a constant time step in mind. THIS FUNCTION IS ...
6     %HARDCODED
7     %WITH A LTI SYSTEM.
8     %
9     % INPUTS:
10    %     Q_k           Matrix of the system noise: cov(v(k))
11    %     R_k1          Matrix of the sensor noise: cov(w(k))
12    %     P_k_k         Matrix of the current system covariance
13    %     X_hat_k_k     Current state estimate
14    %     Z_k1          Measurement of the state at the new time step
15    %
16    % OUTPUTS:
17    %     X_hat_k1_k1   New state estimate
18    %     P_k1_k1       Matrix of the new system covariance
19
20    %%% Begin Kalman Filter %%%
21    % Could use global vars, arguments, or hard-coded matrices. Hard ...
22    % coded is faster.
23
24    H_k1 = eye(3);
25    F_k = eye(3);
26
27    %%% PREDICTION
28    X_hat_k1_k = F_k * X_hat_k_k; % + G_k*u_k; % No inputs
29    P_k1_k = F_k * P_k_k * F_k' + Q_k;
30
31    %%% CORRECTION
32    % Calculate Kalman Gain
33    K_k1 = P_k1_k * H_k1 * pinv(H_k1 * P_k1_k * H_k1' + R_k1);
34    % Calculate error between measurement and prediction
35    Z_k1_tilde = Z_k1 - H_k1 * X_hat_k1_k;
36    % Correct Prediction
37    X_hat_k1_k1 = X_hat_k1_k + K_k1 * (Z_k1_tilde);
38
39    %%% UPDATE COVARIANCE
40    P_k1_k1 = (eye(3) - K_k1 * H_k1) * P_k1_k * (eye(3) - K_k1 * H_k1)' + ...
41        K_k1 * R_k1 * K_k1';
42
43 end
```

Appendix E. Top-level Bootstrapped MATLAB Code

The script `OptMain` is the main function for the optimization algorithm.

```
1 %% Solve the SUAS Problem (min t1)
2 clear; close all; clc; diary off;
3
4 %%% NOTES TO THE READER:
5 % Units are to be in feet (except for 6m CEP 90)
6
7 %%% Create/Find Folder for Trial
8 % SPECIFY TRIAL NAMES
9 TrialType = 'Scenario'; % Could be 'Approach', 'Orbit', 'Trivial'
10 TrialP = 'P1'; % Predefine positions 'P1', 'P2', etc.
11 TrialPsi = 'Psi2'; % Predefine headings 'Psi1', 'Psi2', etc.
12 TrialNFZ = 'MtnAppAnd2'; % Mountain in approach and normal 2
13 TrialAlt = '500'; % z_v in (ft) AGL
14 TrialWind = 'S 15'; % Wind Direction and Speed 'NW 15' or '0'
15 TrialName = [TrialType, '-', ...
16     TrialP, TrialPsi, '-', ...
17     TrialNFZ, '-', ...
18     TrialWind, '-', ...
19     TrialAlt, 'AGL']
20
21 FileDir = ['C:\Users\JohnDoe\Documents\'...
22     'SaveFilesHere'];
23 [s, mess, messid] = mkdir(FileDir, TrialName);
24 % Avoid overwriting files unintentionally
25 if(~strcmp(mess, ''))
26     % Construct a questdlg with three options
27     choice = questdlg('Do you want to overwrite this trial?', ...
28         'CAUTION: File Exists', ...
29         'Overwrite', 'Cancel', 'Cancel');
30     % Handle response
31     switch choice
32     case 'Overwrite'
33         disp('Deleting folder contents to overwrite')
34         % DELETE FOLDER CONTENTS
35         delete([FileDir, '\', TrialName, '\*']);
36     case 'Cancel'
37         error('Stopping trial');
38     end
39 else
40     disp('Trial folder did not previously exist: Continuing');
41 end
42
43 %%% Create Record of Trial in Text Document
44 filename = GenerateRecord('OptMain',...
45     [FileDir, '\', TrialName],...
46     TrialName); % Generate text record of trial
```

```

47
48 %% SETTINGS
49 %%% Set Global Variables for Simulation
50 global V CovReq X_o X_t X_t_g Sv P_k_k_o Q_k z_v tf dt;
51 global Wind_Spd Wind_Dir WindRule;
52 global N RangeSensor options CovReqIter_Stop Num_NFZs;
53 global R_NFZ_1 X_NFZ_1 Y_NFZ_1;
54 global R_NFZ_2 X_NFZ_2 Y_NFZ_2;
55 global R_NFZ_3 X_NFZ_3 Y_NFZ_3;
56 global N_t1 N_t2 t2 IdealOrbRad; % For "MAINTAIN"
57
58 %%% Algorithm Settings:
59 AUDIO = 1 % Set to 1 if audio notifications are ...
    desired
60 t2 = 30 % Buffer time after achieving CovReq
61 tf = t2 + 75 % Initial guess for final time
62 StepTol = 1e-6 % Step tolerance for fmincon
63 fminconAlg = 'interior-point' % 'sqp' or 'interior-point'
64 ExtendRule = 'ratio' % Extends state vector by 'ratio' or 'step'
65 ExtendRatio = 1.25 % Extends by a percentage ...
    (ExtendRule='ratio')
66 ExtendStepLength = 5 % Extends by 5 steps (ExtendRule='step')
67 Num_NFZs = 3 % Used to determine which NFZs are active
68 AC_IC_Ang = pi % Initial angular position (rad)
69 SmartGuess = 1 % Allows for a constant Psi_d_guess
70 Psi_d_guess = 0.02; % Initial control guess (rad/s)
71 % NOTE: t1 + t2 = tf (Min t1 fixed t2)
72
73 %%% Unit Conversions and Constants
74 knots2fps = 1.68781; % 1 kn = 1.68781 ft/s
75 meters2feet = 3.28084; % 1 m = 3.28084 ft
76 naut2feet = 6076.11568; % 1 nm = 6076.11568 ft
77 gravity = 32.174; % ft/s^2
78
79 %%% ICs, Flight Constants, & Limits
80 IdealSlantRange = 2000 % Ideal steady state orbit (ft)
81 X_o = [-1.5*naut2feet;
82     0.25*naut2feet;
83     -pi/4] % Initial UAV Location [x, y, psi]
84 z_v = str2num(TrialAlt) % Constant UAV Altitude AGL (ft)
85 X_t = [0; 0; 0] % Target location [x,y,z]
86 V_knots = 55 % UAV Speed (knots)
87 V = 55*knots2fps; % UAV Speed (ft/s)
88 MaxBankDeg = 30 % Max Bank Angle (deg)
89 MaxBank = deg2rad(MaxBankDeg); % Max Bank Angle (rad)
90 SlantRangeMin = 1500 % Min slant range to avoid detection (ft)
91
92 %%% Sensor Specs
93 OperError = [0; 0; 0] % Simulate operator's initial input error.
94 RangeSensor = 0 % Set to 1 if range sensor is available
95 SensorRate = 1.00 % Max frequency of sensor readings in Hz
96 % Note about Sv: The numbers are RMS values, do they need to be squared?

```

```

97 Sv = diag([deg2rad(0.15+0.008103),... % Azimuth Covariance (rad)
98           deg2rad(0.035+0.008103),... % Elevation Covariance (rad)
99           30]) % Range Covariance (ft)
100 Q_k = 1*eye(3) % System Noise (Low for stationary target)
101
102 %%% Dynamic NFZ aircraft and sensor specs
103 AC_Alt = 10000 % (ft)
104 AC_Rad = 6600 % (ft)
105 AC_ktas = 220; % (knots)
106 NFZ_SpreadDeg = 5 % Half angle (deg)
107
108 %%% Targeting Requirements
109 Z_alpha2_90 = 1.645 % To be multiplied by standard deviation
110 CovReq = 6*meters2feet/Z_alpha2_90 % Covariance Component Maximum
111
112 %%% Wind Settings
113 [Direction, Magnitude] = strtok(TrialWind);
114 switch Direction
115     case '0'
116         Wind_Dir = 0;
117         Wind_SpdKn = 0;
118     case 'E'
119         Wind_Dir = 0;
120     case 'NE'
121         Wind_Dir = pi/4;
122     case 'N'
123         Wind_Dir = pi/2;
124     case 'NW'
125         Wind_Dir = 3*pi/4;
126     case 'W'
127         Wind_Dir = pi;
128     case 'SW'
129         Wind_Dir = 5*pi/4;
130     case 'S'
131         Wind_Dir = 3*pi/2;
132     case 'SE'
133         Wind_Dir = 7*pi/4;
134     otherwise
135         error('Wind is incorrectly defined');
136 end
137 if ~strcmp(WindRule, 'none')
138     WindRule = 'constant';
139     if(str2num(Direction)==0)
140         Wind_SpdKn = 0;
141     else
142         Wind_SpdKn = str2num(Magnitude);
143     end
144 end
145
146 %%% Calculated Simulation Values and Settings
147 IdealOrbRad = sqrt(IdealSlantRange^2 - z_v^2); % (ft)
148 Wind_Spd = Wind_SpdKn*knots2fps; % Wind Speed (ft/s)

```

```

149 R_turn_min = (V^2)/(gravity*tan(MaxBank)); % Min Turn Radius (ft)
150 psi_d_max = V/R_turn_min; % Max turn rate(rad/sec)
151 N = ceil(SensorRate*tf); % Number of points for discretization
152 N_t1 = floor((tf-t2)*SensorRate); % Index of t1
153 N_t2 = N-N_t1; % Steps after t1
154 dt = 1/SensorRate; % Fixed time step
155 X_t_g = X_t + OperError; % Initial Guess (could have some error)
156 % numStates = 3; % [ x, y, psi] NOT SURE IF STILL NEEDED
157 P_k_k_o = 100*eye(3) % Initial State Covariance (High)
158 CovReqIter_Stop = -1; % This is a flag.
159
160 %% Specify No-Fly-Zone Parameters
161 % Detection NFZ
162 R_NFZ_1 = sqrt(SlantRangeMin^2 - (z_v-X_t(3))^2) % (ft)
163 R_NFZ_1 = R_NFZ_1*ones(1,1000); % (ft)
164 X_NFZ_1 = X_t(1)*ones(1,1000); % (ft)
165 Y_NFZ_1 = X_t(2)*ones(1,1000); % (ft)
166
167 % Air Craft Sight-Line NFZ
168 AC_Spd = AC_ktas*knots2fps; % (ft/s)
169 NFZ_Spread = deg2rad(NFZ_SpreadDeg); % (rad)
170 AC_LookAngle = atan(AC_Rad/AC_Alt); % Between vert and targ (rad)
171 R_NFZ_2 = (AC_Alt-z_v)*(sin(AC_LookAngle+NFZ_Spread) - ...
172 sin(AC_LookAngle-NFZ_Spread))/2 % Major axis of NFZ ellipse (ft)
173 R_NFZ_2 = R_NFZ_2*ones(1,1000); % Array
174 AC_AngVel = AC_Spd/AC_Rad; % (rad/s)
175 AC_NFZ_Rad = AC_Rad - ...
176 (AC_Alt-z_v)*sin(AC_LookAngle); % Radius of NFZ orbit (ft)
177 NFZ_2_theta = linspace(AC_IC_Ang,...
178 AC_AngVel*dt*1000 + AC_IC_Ang,...
179 1000);
180 X_NFZ_2 = AC_NFZ_Rad*cos(NFZ_2_theta) + X_t(1); % (ft)
181 Y_NFZ_2 = AC_NFZ_Rad*sin(NFZ_2_theta) + X_t(2); % (ft)
182
183 % Obstruction NFZ
184 R_NFZ_3 = 2000
185 R_NFZ_3 = R_NFZ_3*ones(1,1000); % NFZ_3 radius (ft)
186 X_NFZ_3 = -1.0*naut2feet
187 X_NFZ_3 = X_NFZ_3*ones(1,1000); % (ft)
188 Y_NFZ_3 = 1000
189 Y_NFZ_3 = Y_NFZ_3*ones(1,1000); % (ft)
190
191 %% Algorithm
192 %% Set initial state vector
193 % NOTE: starting point: X_0 = [x, y, psi-d]
194 disp('-----');
195 disp('Begin Optimization Algorithm');
196
197 if(SmartGuess==1)
198 % Initialize with constant psi-d vector if desired
199 x0 = ones(N,1)*Psi_d_guess;
200 else

```

```

201     % Initialize with zero psi_d_v vector
202     x0 = zeros(N,1);
203 end
204
205 %%% Begin Bootstrapping
206 % Begin Timer
207 tic
208 while true
209     while true
210         %%% Set Bounds on states and controls
211         LB = -psi_d_max*ones(N,1);           % psi_d_v min
212         UB = psi_d_max*ones(N,1);           % psi_d_v max
213
214         %%% Set fmincon options
215         %'Algorithm', 'sqp',...
216         options = optimoptions('fmincon',...
217             'Algorithm', fminconAlg,...
218             'OutputFcn', @PlotCurrentGuess_SmartStop_t1_t2,...
219             'TolCon', 1e-6,...
220             'Display', 'iter',...
221             'MaxFunEvals', 1e7,...
222             'MaxIter', 1e7,...
223             'ScaleProblem', 'obj-and-constr',...
224             'StepTolerance', StepTol);
225
226         % Call MATLAB's fmincon
227         [xopt,fval,exitflag,output] = fmincon(...
228             @(x) CostFun_MinT1Cov_Ctrl_Orb(x),...
229             x0,[],[],[],[], LB, UB,...
230             @(x) Constraints_NFZ(x),options);
231
232         % Print file run time to screen and diary
233         fprintf('Time To Optimize: %s\n',duration(0,0,toc));
234
235         % View fmincon output
236         output
237         printExitMessage( exitflag );
238         %%% Check if solution is valid
239         if exitflag==-2
240             % If this runs, StepTol was too big
241             warning('BOOTSTRAPPING ISSUE');
242             printExitMessage(exitflag);
243             fprintf('N = %d\n',N);
244             warning('INCREASE StepTol???');
245         elseif ~any(exitflag==[1 2 -1])
246             % If this runs, the solver ran into an issue.
247             warning('BOOTSTRAPPING FAILED');
248             printExitMessage(exitflag);
249             fprintf('N = %d\n',N);
250             break;
251         end
252

```

```

253     %% Check if Bootstrapping Finished
254     if CovReqIter_Stop==N_t1
255         disp('BOOTSTRAPPING COMPLETED: Smart Stop');
256         break;
257     end
258
259     %% If bootstrapping not complete: prepare for next iteration
260     fprintf('%d iterations not optimal:\n',N);
261     disp('Bootstrapping for next iteration');
262
263     % Pull out optimal vectors
264     psi_d_opt = x_opt(1:N);
265     [ x_opt, y_opt, ~ ] = getPathFromPsi_d( psi_d_opt );
266
267     % Get histories of kalman filter
268     [ P_k_k_hist, ~, ~ ] = GetKalmanHistory(x_opt,y_opt);
269     P_k_k_maxes = max(P_k_k_hist,[],2);
270
271     %% Select N for next iteration
272     old_N = N;
273     N_t1 = find(P_k_k_maxes<=CovReq,1); % First step CovReq is ...
        attained
274
275     %% Check if covariance goal not attained
276     if isempty(N_t1)
277         disp('Covariance requirement not attained with N steps');
278         if strcmp(ExtendRule,'step')
279             N = old_N + ExtendStepLength;
280         elseif strcmp(ExtendRule,'ratio')
281             N = ceil(old_N*ExtendRatio);
282         else
283             error('ExtendRule NOT VALID');
284         end
285
286         fprintf('Increasing N from %d to %d by ...
        %s.\n',old_N,N,ExtendRule);
287
288         %% Append new steps to currently optimal state vector
289         x0 = zeros(N,1);
290         x0(1:old_N) = psi_d_opt;    % Optimal psi_d followed by ...
            zeros
291         tf = N*dt;
292         N_t1 = N-N_t2;
293     elseif (N_t1+N_t2>N)
294         disp('Covariance Goal attained in more than N_t1+N_t2 ...
        steps. ');
295
296         % Consider this???
297         N = N_t1 + N_t2;
298         x0 = zeros(N,1);
299         x0(1:old_N) = psi_d_opt(1:old_N);
300         tf = N*dt;

```

```

301         fprintf('Increasing N from %d to %d by %s.\n',...
302                 oldN,N, 'N.t2 Buffer');
303     else
304         % NOTE: THIS CAUSES AN ISSUE WITH THE COMPLEX COST FUNCTION?
305         disp('Covariance Goal attained in less than N.t1+N.t2 ...
306               steps. ');
307         disp('Decreasing N to smallest valid size');
308         N = N.t1 + N.t2;
309         x0 = zeros(N,1);
310         x0(1:N) = psi_d_opt(1:N); % Trimmed Optimal psi_d
311         tf = N*dt;
312     end
313     %%% Begin next iteration
314 end
315 %%% Verify that solution is valid
316 % Extract optimal states and control
317 psi_d_opt = xopt(1:N);
318 [ x_opt, y_opt, psi_opt ] = getPathFromPsi_d( psi_d_opt );
319 tf_optimal = N.t1/SensorRate;
320 % tf = N/SensorRate; % I don't think this is needed.
321 [g, h] = Constraints_NFZ(xopt);
322
323 % Get histories of kalman filter
324 [ P_k_k_hist, Cov_xyz_hist, X_t_hist ] = ...
325     GetKalmanHistory(x_opt,y_opt);
326
327 % View Control Vector in Plot
328 figure(100)
329 plot(psi_d_opt);
330 title('Control');
331 %%% Bootstrapping Complete
332 % Notify Optimization Algorithm Completion With Sound
333 if(AUDIO)
334     if(exitflag==1 || exitflag==-1) %#ok<UNRCH>
335         load handel;
336         player = audioplayer(y, Fs);
337         play(player);
338     else
339         load gong.mat;
340         gong = audioplayer(y, Fs);
341         play(gong);
342         pause(5);
343         play(gong);
344     end
345 end
346 disp(StepTol);
347 % Allow user to decrease StepTol if desired
348 choice = questdlg('Decrease StepTol?', ...
349                 'StepTol Issue', ...
350                 '1e-1', '1e-2', 'No', '1e-1');
351 % Handle response

```

```

351     switch choice
352         case '1e-1'
353             disp('Decreasing StepTol and Bootstrapping')
354             StepTol = StepTol/10
355             x0 = xopt;
356         case '1e-2'
357             disp('Decreasing StepTol and Bootstrapping')
358             StepTol = StepTol/100
359             x0 = xopt;
360         case 'No'
361             disp('StepTol is unchanged')
362     %         break;
363     end
364     % Allow user to change algorithm if desired
365     choice = questdlg('Select algorithm:', ...
366         'Algorithm', ...
367         'SQP', 'Interior-Point', 'Stop', 'Interior-Point');
368     switch choice
369         case 'SQP'
370             disp('Using SQP')
371             fminconAlg = 'sqp'
372         case 'Interior-Point'
373             disp('Using Interior-Point')
374             fminconAlg = 'interior-point'
375         case 'Stop'
376             disp('Finished Optimizing')
377             break
378     end
379 end
380
381 %% Create Plots of the Optimal Results
382 % CODE NOT SHOWN
383
384 %% End of script functions
385 diary off;
386 CloseRecord(filename);

```

Appendix F. Point-Mass Model MATLAB Code

The function `getPathFromPsi_d` calculates the trajectory of the UAV given a control vector. The function accounts for a constant wind in the point-mass model

```
1 function [ x_v, y_v, psi_v ] = getPathFromPsi_d( psi_d_v )
2     %getPathFromPsi_d Given the initial state and a psi_d history,
3     %calculates the full state vector for a trajectory.
4     %
5     %   INPUTS:
6     %       psi_d_v       1xN vector from psi_d_v0 to psi_d_vf
7     %
8     %   OUTPUTS:
9     %       x_v           Nx1 vector of the vehicles east positions
10    %       y_v           Nx1 vector of the vehicles north positions
11    %       psi_v          Nx1 vector of the vehicles heading angles
12
13    %%% Declare necessary global variables
14    global V dt N X_o Wind_Dir Wind_Spd
15
16    %%% Initialize x_v, y_v, psi_v to be the proper length
17    x_v = [X_o(1); zeros(N-1,1)];      % East position of UAV
18    y_v = [X_o(2); zeros(N-1,1)];      % North position of UAV
19    psi_v = [X_o(3); zeros(N-1,1)];    % Heading angles of UAV
20
21    %%% Loop through and calculate psi_v, x_v, and y_v for each step
22    for i=1:N-1
23        % psi_v propagation
24        psi_v(i+1) = psi_v(i) + psi_d_v(i)*dt;
25
26        % x_v propagation
27        x_v(i+1) = x_v(i) + V*dt*cos(psi_v(i)) + ...
28            Wind_Spd*dt*cos(Wind_Dir);
29
30        % y_v propagation
31        y_v(i+1) = y_v(i) + V*dt*sin(psi_v(i)) + ...
32            Wind_Spd*dt*sin(Wind_Dir);
33    end
34 end
```

Bibliography

- [1] M. T. Grabbe, B. M. Hamschin, and A. P. Douglas, “A measurement correlation algorithm for Line-of-Bearing geo-location,” in *2013 IEEE Aerospace Conference*, March 2013, pp. 1–8, Big Sky, MT.
- [2] S. Ross, “Stochastic real-time optimal control for bearing-only trajectory planning,” Ph.D. dissertation, Air Force Institute of Technology, 2011.
- [3] A. Inc., “Media gallery: Raven,” 2016, accessed: 2016-10-26. [Online]. Available: https://web.archive.org/web/20161026162833/https://www.avinc.com/media_center/unmanned-aircraft-systems/raven
- [4] U. S. Technology, “Ara demonstrates search and rescue capabilities of nighthawk micro UAS,” <https://web.archive.org/web/20140716221506/http://www.unmannedsystemstechnology.com/2013/12/ara-demonstrates-search-and-rescue-capabilities-of-nighthawk-micro-uas/>, accessed: 2016-10-20.
- [5] I. Applied Research Associates, “Nighthawk — micro UAV,” 2015, accessed: 2016-10-24. [Online]. Available: <https://web.archive.org/web/20151231204445/http://forcepro.ara.com/products/force-protection/nighthawk-micro-uav>
- [6] B. K. Funk, R. J. Bamberger, J. D. Barton, A. K. Carr, J. C. Castelli, A. B. Cox, D. G. DREWRY, S. H. Popkin, and A. S. Watkins, “Enabling technologies for small unmanned aerial system engagement of moving urban targets,” *Johns Hopkins APL technical digest*, vol. 31, no. 2, pp. 150–166, 2012.
- [7] Officer.com, “Aras advanced micro-UAV certified for further flights,” 2015, accessed: 2016-10-24. [Online]. Avail-

able: https://web.archive.org/web/20170301145121/http://www.officer.com/press_release/11290269/aras-advanced-micro-uav-certified-for-further-flights

- [8] Airforce-technology.com, “Maveric mini unmanned aerial system (MUAS), United States of America,” 2016, accessed: 2016-10-26. [Online]. Available: <https://web.archive.org/web/20161002143218/http://www.airforce-technology.com/projects/maveric-small-unmanned-aerial-system/>
- [9] M. Pachter, N. Ceccarelli, and P. R. Chandler, “Vision-based target geolocation using micro air vehicles,” *Journal of guidance, control, and dynamics*, vol. 31, no. 3, pp. 597–615, 2008.
- [10] M. C. Deans, “Bearings-only localization and mapping,” Ph.D. dissertation, Carnegie Mellon University, 2005.
- [11] L. DO Q, “Numerically efficient methods for solving least squares problems,” 2012, Citeseer.
- [12] S. M. Ross, R. G. Cobb, and W. P. Baker, “Stochastic real-time optimal control for bearing-only trajectory planning,” *International Journal of Micro Air Vehicles*, vol. 6, no. 1, pp. 1–27, 2014.
- [13] T. A. Sands, “Lecture notes in nonlinear controls,” July 2016, Air Force Institute of Technology.
- [14] M. Verhaegen and P. Van Dooren, “Numerical aspects of different implementations,” *IEEE Transactions on Automatic Control*, vol. 31, no. 10, pp. 907–917, 1986.
- [15] C. S. Horstmann, *Big Java: Compatible with Java 5, 6 and 7*. John Wiley & Sons, 2009.

- [16] H. Satz and T. H. Kerr, “Comparison of batch and Kalman filtering for radar tracking,” DTIC Document, Tech. Rep., July 2001, ADPO11192.
- [17] T. R. Jorris and R. G. Cobb, “Multiple method 2-D trajectory optimization satisfying waypoints and no-fly zone constraints,” *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 3, pp. 543–553, 2008.
- [18] W. Mathworld, “Point-line distance–3-dimensional,” 2016, accessed: 2016-11-09. [Online]. Available: <https://web.archive.org/web/20160523131350/http://mathworld.wolfram.com/Point-LineDistance3-Dimensional.html>
- [19] MathWorks, “Matlab functions,” 2016, accessed: 2016-11-10. [Online]. Available: <https://www.mathworks.com/help/matlab/functionlist.html>
- [20] W. Nelson, “Use of circular error probability in target detection,” DTIC Document, Tech. Rep., May 1988, ADA199190.
- [21] N. Jodeh, “Optimal UAS assignments and trajectories for persistent surveillance and data collection from a wireless sensor network,” Ph.D. dissertation, Air Force Institute of Technology, 2015.
- [22] A. H. Barr, “Superquadrics and angle-preserving transformations,” *IEEE Computer graphics and Applications*, vol. 1, no. 1, pp. 11–23, 1981.

REPORT DOCUMENTATION PAGE					<i>Form Approved</i> OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.						
1. REPORT DATE (DD-MM-YYYY) 6 March 2017		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From — To) May 2016 — 6 March 2017		
4. TITLE AND SUBTITLE OPTIMAL UAV PATH PLANNING WITH DYNAMIC NO-FLY-ZONES FOR TARGET GEOLOCATION USING LINE-OF-BEARING MEASUREMENTS AND KALMAN FILTERING				5a. CONTRACT NUMBER		
				5b. GRANT NUMBER		
				5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Heidlauf, Peter, Second Lieutenant				5d. PROJECT NUMBER 17V-152		
				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-MS-17-M-262		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/RQQA Att: Mr. Paul Fleitz 2210 8th Street WPAFB OH 45433 Email: paul.fleitz@us.af.mil				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RQ		
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION / AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.						
13. SUPPLEMENTARY NOTES						
14. ABSTRACT An optimal trajectory generation algorithm is needed for a Tactical Off-Board Sensing (TOBS) UAV to maximize the utility of its endurance while providing actionable target coordinates to other assets. A method of path optimization was developed to geolocate a stationary target using Line-of-Bearing measurements and to reach a prescribed orbit about the target in minimum time. Dynamic No-Fly-Zones (NFZ) were enforced, and different winds and initial conditions were simulated. A bootstrapping method was developed to approach the true minimum final time solution by iteratively minimizing the target covariance at a specified time step. A Kalman filter estimated the target position to a 6 meter CEP 90% requirement. Real-world scenarios were simulated for a candidate TOBS airframe and optimized to determine mission performance, trajectory characteristics, and to provide recommendations for the TOBS program. Simulations showed targeting time decreases of up to 19% dependent on wind conditions, and by up to 10% dependent on NFZ-relative launch positioning. Targeting requirements were met during all deviations from target orbit.						
15. SUBJECT TERMS						
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. Cobb, AFIT/ENY	
U	U	U	UU	126	19b. TELEPHONE NUMBER (include area code) (937) 255-3636; Richard.Cobb@afit.edu	