©Copyright 2014
Kevin K. Ueunten

Modeling Aircraft Position and Conservatively Calculating Airspace Violations for an Autonomous Collision Awareness System for Unmanned Aerial Systems

Kevin K. Ueunten

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2014

Reading Committee:

Dr. Juris Vagners, Chair

Dr. Christopher Lum

Program Authorized to Offer Degree:
William E. Boeing Department of Aeronautics and Astronautics

University of Washington

Abstract

Modeling Aircraft Position and Conservatively Calculating Airspace Violations for an Autonomous Collision Awareness System for Unmanned Aerial Systems

Kevin K. Ueunten

Chair of the Supervisory Committee:
Professor Emeritus Dr. Juris Vagners
William E. Boeing Department of Aeronautics and Astronautics

With the scheduled 30 September 2015 integration of Unmanned Aerial System (UAS) into the national airspace, the Federal Aviation Administration (FAA) is concerned with UAS capabilities to sense and avoid conflicts. Since the operator is outside the cockpit, the proposed collision awareness plugin (CAPlugin), based on probability and error propagation, conservatively predicts potential conflicts with other aircraft and airspaces, thus increasing the operator's situational awareness. The conflict predictions are calculated using a forward state estimator (FSE) and a conflict calculator. Predicting an aircraft's position, modeled as a mixed Gaussian distribution, is the FSE's responsibility. Furthermore, the FSE supports aircraft engaged in the following three flight modes: free flight, flight path following and orbits. The conflict calculator uses the FSE result to calculate the conflict probability between an aircraft and airspace or another aircraft. Finally, the CAPlugin determines the highest conflict probability and warns the operator. In addition to discussing the FSE free flight, FSE orbit and the airspace conflict calculator, this thesis describes how each algorithm is implemented and tested. Lastly two simulations demonstrates the CAPlugin's capabilities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

CAPLUGIN: collision awareness plugin

DCM: direction cosine matrix

FAA: Federal Aviation Administration

FSE: forward state estimator

GPS: global positioning satellite

ICOMC2: Insitu Common Open-mission Management Command and Control System

JCATI: Joint Center for Aerospace Technology Innovation

NAS: national airspace system

UAS: unmanned aerial system

ACKNOWLEDGMENTS

I would like to recognize the many people and organizations who supported me on this research project. Thank you for the Washington's Joint Center for Aerospace Technology Innovation (JCATI) for funding this project, the JCATI director, Dr. Mehran Mesbahi, and the JCATI's program manager, Patrick Gibbs. I am grateful for Insitu's contributions and in-kind support to this project. And I would like to recognize the following Insitu members for their direct involvement with this research project: Dr. Rolf Rysdyk, Keith Ketring, Andrew Hayes, Darcy Davidson, Jim Miller, Aaron High, Amy Arbeit, Jeremy Tate, Jonny Polivka, and Christy Grimm. I appreciate the following guest speakers for sharing their expertise: Art Crowe, Nathaniel Guy, Matt McCully and Dr. Dick Newman. I would like to recognize Laura Dorsey for handling the intellectual property and technology transfer of this research. Lastly, I am grateful for the research team, who have worked countless hours developing this algorithm: Dr. Christopher Lum, Dr. Juris Vagners, Al Creigh, Keisuke Tsujita, Madison Peck, Richard Fukutome, Matthew Davis, Bao Le, Henry Qin, Brian Chang, Taylor Campbell, Daniel Ablog, Federico Alvarez, Noel Kimber, Dai Tsukada, Robert McSwain, Seunghyun Ko, Ryan Valach, Aleksandr Tereshchenkov, Justin Yantus and John Marshall. Finally, thank you Dr. Christopher Lum for providing guidance throughout this project and teaching me C#.

DEDICATION

I want to dedicate this thesis to my teachers,who have inspired me to continue learning.

Chapter 1

INTRODUCTION

Unmanned Aerial Systems (UAS) provide many societal benefits, such as search and rescue operations [1], [2], [3] aiding in agriculture development, predicting weather or monitoring pipelines. However to fully utilize these benefits the National Airspace System (NAS) must be open for UAS operations. Currently, the Federal Aviation Administration (FAA) is developing policy to safely integrate UASs into the NAS [4]. In 2013 the FAA released *Integration of Civil Unmanned Aircraft Systems Into National Airspace System*, which describes the FAA's concern with UAS operations in the NAS and their plan for UAS integration. In section 332 section A, paragraph 3, the FAA states, "The plan required under paragraph (1) shall provide for the safe integration of civil unmanned aircraft systems into the national airspace system as soon as practicable, but not later than September 30, 2015" [5]. To meet the September 2015 deadline, UASs must demonstrate many capabilities to include collision avoidance. The FAA defines collision avoidance as "The Sense and Avoid system function where the UAS takes appropriate action to prevent an intruder from penetrating the collision volume. Action is expected to be initiated within a relatively short time horizon before closest point of approach. The collision avoidance function engages when all other modes of separation fail" [5]. However since UASs do not have pilots in the cockpit, the FAA's sense and avoid concept for UAS is slightly modified and defined as "The capability of a UAS to remain well clear from and avoid collisions with other airborne traffic. Sense and Avoid provides the functions of self-separation and collision avoidance to establish an analogous capability to "see and avoid" required by manned aircraft" [5].

Although the UAS operator's primary responsibility is collision avoidance, the operator juggles many other primary responsibilities, such as flying the mission [6], [7] and operating the payload. By managing numerous duties, the operator may become task saturated and lose focus on other responsibilities. Algorithms have been developed to improve the operator and UAS interaction [8], [9] but a collision warning system has not been specifically built. Also prior studies [10], [11], [12],

[13], [14] investigated the cost and risk with flying UAS missions and created a risk analysis tool that factors in where the UAS is operating, potential collateral damage and population densities. Therefore developing an automated collision warning system reduces an operator's work load while maintaining situational awareness, thus decreasing the risk associated with operating UAS in the national airspace. Unlike collision avoidance algorithms researched by [15], [16], the proposed collision awareness system does not autonomously avoid collisions, rather the system only warns the operator, who then takes appropriate actions to avoid the conflict.

Today, manned aircraft uses a traffic collision avoidance system (TCAS) to automatically maintain separation distances and increase the pilot's situational awareness [17]. TCAS is highly regarded by the aviation community and pilots are instructed to follow TCAS warnings to prevent collisions [18]. Currently, a system similar to TCAS has not been specifically implemented for UASs. The algorithms and systems outlined are aimed to reduce operator workload with respect to current and impending conflicts. Unlike TCAS, the proposed system has default settings which the operator can modify to vary the sensitivity levels. Furthermore, the system's predictive threat detection capabilities allow for multiple-threat resolution and restricted airspace warnings. These systems can be integrated into existing UAS ground station software such as the Insitu Common Open Mission Command and Control (ICOMC2) [19] to highlight potential future conflicts with the operator's vehicle and other air traffic or restricted airspace.

1.1 Research Partnership

The collision awareness plugin (CAPlugin) development project was funded by the Joint Center for Aerospace Technology Innovation (JCATI) a Washington State initiative that partners universities in Washington with Washington aerospace companies. A university and industry partnership allows students to learn more about Washington's aerospace industries and provides real world experience by solving industry level problems [20]. Our industry partner is Insitu Inc., a Bingen, WA based UAS company. Insitu has developed their own ground station, Insitu Common Open-mission Management Command and Control System (ICOMC2) [19], and hopes to increase the operator's situational awareness by integrating a CAPlugin with ICOMC2. Throughout this one year project, which started in August 2013 and ended on June 30 2014, monthly meetings were held between

Insitu and the University of Washington's research team for both progress checks and feedback. The research team is comprised of many people to include: Dr. Christopher Lum, principal investigator, two graduate students, two alumnus and numerous undergraduate engineering students from the class of 2014, 2015, 2016, who study various engineering disciplines. Working on an academic project that is evaluated and reviewed by industry and potentially be integrated into real world system is a unique opportunity and satisfying experience. Overall this partnership provided insight into managing a software development project and developing production level code.

1.2 Previous Work

Collision awareness and avoidance algorithms is an actively researched topic where the collision awareness algorithm is created based for either a specific vehicle or an onboard sensors. For instance, Domenico Accardo from the University of Naples and his research team developed a collision awareness algorithm using radar signals sent from the aircraft [21]. Similarly a US patent by David Duggan involves sensors on a UAS to track and predict other aircraft positions. Once a collision is imminent, the UAS autonomously avoids the conflict [22]. Other researchers, like Chunbo Luo from the University of Ulster, Coleraine, U.K. and his team, have used Kalman filters and their variations to predict other aircraft's trajectories [23]. Other collision awareness algorithms focus on cooperative aircraft, a system where aircraft send information between each other. Professor Chin E. Lin from the National Cheng Kung University, Tainan, Taiwan developed a collision avoidance algorithm for a UAS and helicopter cooperative system. This algorithm requires information from the helicopter [24]. Another example is a UAS cooperative system used to prevent conflicts as described in Venanzio Cichella's *Trajectory Generation and Collision Avoidance for Safe Operation of Cooperating UAVs* paper [25]. Most collision awareness research is provided with known aircraft dynamic models or information from sensors on board, however, the proposed CAPlugin utilizes information received only by the ground station. Since the CAPlugin needs to provide conflict probabilities quickly, a simple closed form solution is desired. Also the CAPlugin is a modular plugin for ICOMC2, therefore requiring minimal memory usage. Furthermore, the CAPlugin predicts aircraft positions when an aircraft is engaged in free flight, following a flight path or orbiting. Another requirement is the CAPlugin predicts both aircraft conflicts and airspace violations.

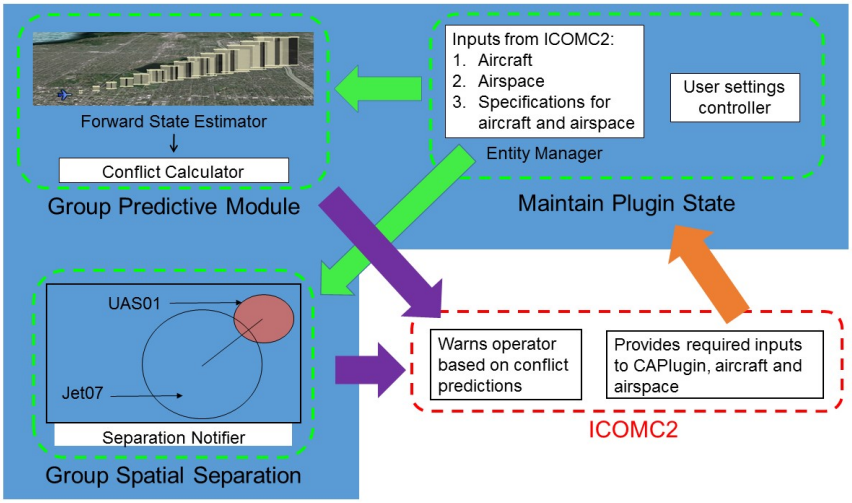

Figure 1.1: CAPlugin architecture and the interaction between the CAPlugin and ICOMC2

The collision awareness plugin (CAPlugin) architecture is comprised of three modules: maintain plugin states, group predictive module, and group spatial separation. Furthermore this shows the interaction between the CAPlugin and Insitu's ICOMC2.

1.3 Collision Awareness Plugin Architecture

ICOMC2 provides the CAPlugin with the required inputs. Using these inputs the CAPlugin returns potential conflict information to ICOMC2, which appropriately warns the operator. The CAPlugin contains three main components: maintaining plugin state, group predictive module, and group spatial separation. When ICOMC2 sends information like aircraft information (position, velocity, flight mode, etc) or restricted airspace parameters to the CAPlugin, the "maintain state module" transforms this data into the inputs used by the group predictive module and group spatial separation. The group predictive module first calculates an aircraft's position via the forward state estimator. Given the forward state estimator results, the conflict calculator determines the conflict probability, the conflict location and when the conflict occurs. The group spatial separation uses the separation notifier to determine if there are any conflicts occurring currently. Results from both the group predictive module and the group spatial separation module are returned to ICOMC2. Figure 1.1 illustrates the collision awareness architecture.

1.4 Thesis Overview

This paper describes the Forward State Estimator (FSE) free flight, FSE orbit and the airspace conflict calculator. After explaining and demonstrating each component with a plethora of worst case scenarios, we present a discussion about implementing these three algorithms into the modular plugin. Lastly, the CAPlugin's capabilities are illustrated with two simulations.

1.4.1 Forward State Estimator

The forward state estimator (FSE) conservatively predicts[1] an aircraft's future position in the 3D Cartesian space. Since there is an error associated with an aircraft's position, the position is modeled stochastically as a mixed Gaussian distribution, a 2D distribution in the xy-plane and a 1D distribution in the z-direction. Furthermore this mixed distribution is a reasonable assumption because the GPS error in the xy-direction is different than the z-direction. The FSE models every aircraft as a 3D point mass, where the distribution's mean is determined by 3D kinematic motion and the covariance is determined from a continuous time error propagation model. Furthermore, the FSE supports three different flight modes: free flight, flight path and orbit. An aircraft engaged in following a flight path or an orbit has known intent, thus the FSE result has reasonable constraints based on the flight path or the orbit. An aircraft with unknown intent is in free flight, which assumes the aircraft continues to fly at the current heading and speed.

Required Inputs

For all flight modes the FSE requires the aircraft's current position and velocity and aircraft specific errors. The parameter errors are the aircraft's GPS error in the xy-plane (σ_{x0} and σ_{y0}) and in the z-direction (σ_{z0}), speed error (σ_s), climb angle error (σ_ϕ) and heading angle error (σ_θ). Furthermore the error provided is at the 95% confidence level. The standard deviation for each parameter variable is calculated using a standard normal distribution (Eq.1.1). The standard deviation is required to calculate the distribution's covariance and variance.

---

[1] A conservative estimate is the region in 3D space which contains the mixed Gaussian distribution's 95% or operator specified confidence level.

$$\sigma = \frac{95\% \text{ confidence error}}{1.96} \tag{1.1}$$

To illustrate this let the 95% confidence error for position be 5m $\pm$ 3m. Thus the 95% confidence error is 3m and

$$\sigma = \frac{3\text{m}}{1.96} = 1.53\text{m} \tag{1.2}$$

Additionally, the FSE orbit requires information about the orbit's center, radius and orbit direction.

1.4.2 Conflict Calculator

The conflict calculator requires an FSE result, aircraft specific information and airspace properties to determine the conflict probability between a perspective entity, the operator controlled aircraft, and an aircraft or an airspace. Integrating the Gaussian distribution yields the conflict probability. By over approximating conflict probabilities, conservative results are obtained.

Chapter 2

FORWARD STATE ESTIMATOR FREE FLIGHT

The FSE free flight is the default flight mode for all aircraft with unknown intent and conservatively overestimates an aircraft's future position. For this paper a conservative estimate is the region which contains the mixed Gaussian distribution with 95% confidence[1]. A 3D kinematic point mass model determines the distribution's mean, while a continuous time second order error propagation model calculates the covariance.

2.1 Mean Calculation

An aircraft engaged in free flight continues to fly at the current heading, climb angle and speed. The climb angle (ϕ) and θ [2] are calculated based on the current velocity's components. Figure 2.1 illustrates how θ and ϕ are defined. The speed (s) is the current velocity's magnitude and is essential to calculate the mixed Gaussian distribution for both the FSE free flight and orbit. s is key in calculating both the FSE means, covariances and variances.

Applying these three parameters an aircraft's mean position is

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + st \begin{bmatrix} \cos(\phi)\cos(\theta) \\ \cos(\phi)\sin(\theta) \\ sin(\phi) \end{bmatrix} \tag{2.1}$$

where $\begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}^{\mathrm{T}}$ is the initial position and t is the prediction time.

---

[1] This confidence level is operator specified with the default value at 95%.

[2] θ is similar to a heading angle but the angle is defined differently. θ has the same definition as polar coordinate angles.

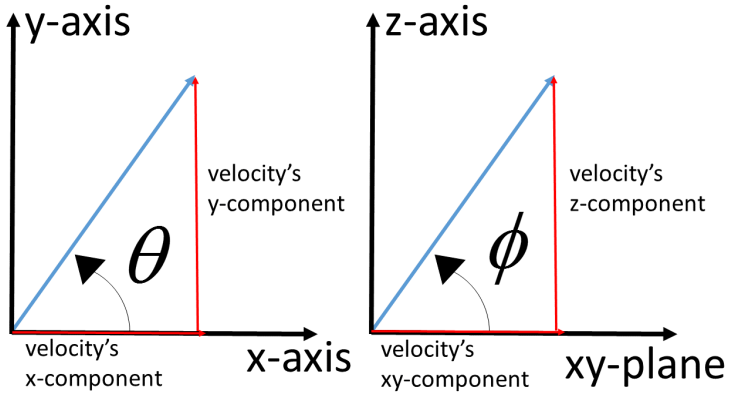

Figure 2.1: FSE free flight's θ and ϕ description

θ is the angle the aircraft travels in the xy-plane. ϕ is the aircraft's climb angle, which is the angle between the xy-plane and the z-axis.

2.2 Covariance Calculation

The covariance is calculated using a continuous time error propagation model in the body frame. A body frame system (Figure 2.3) is appropriate because the body frame's +x-axis aligns with θ.

The aircraft's motion in the body frame is described by Eq.2.2.2. To account for the worst case scenario the s in Eq.2.1 is used in Eq.2.2.2. For example let the velocity be $[2\frac{\text{m}}{\text{sec}},\ 2\frac{\text{m}}{\text{sec}},\ 1\frac{\text{m}}{\text{sec}}]^{\text{T}}$ thus the speed is $3\frac{\text{m}}{\text{sec}}$. Therefore the speed in the body's x-axis direction is actually $2\frac{\text{m}}{\text{sec}}$ but $3\frac{\text{m}}{\text{sec}}$ will be used. Since the actual speed in each body axis is less than the aircraft's speed, the body motion equations will always capture the worst case scenarios and yield conservative results. Also since the body frame is aligned with the aircraft's motion, θ_{body} and ϕ_{body} are 0°.

$$\begin{bmatrix} x_{body}(t) \\ y_{body}(t) \\ z_{body}(t) \end{bmatrix} = \begin{bmatrix} x_{body}(0) \\ y_{body}(0) \\ z_{body}(0) \end{bmatrix} + st \begin{bmatrix} \cos(\theta_{body}) \\ \sin(\theta_{body}) \\ \cos(\phi_{body}) \end{bmatrix} \tag{2.2}$$

Once the body covariance is calculated a direction cosine matrix (DCM) with rotation angle θ determines the covariance in the Cartesian system. However, there are two error propagation models that can be used: a first order error propagation method and a second order error propagation method.

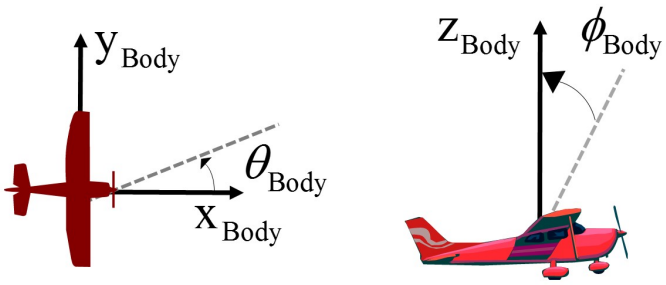

Figure 2.3: Body coordinate system description

This is the body coordinate system where the x-body axis starts from the aircraft's center to the aircraft's nose and is aligned with θ and the y-body axis begins at the aircraft's center and ends at the left wingtip. Furthermore z-body axis is the cross product of the x-body axis and y-body axis. θ_{Body} is the angular deviation from the x-body axis. Similarly the ϕ_{Body} is the angular deviation from the z-body axis

The next section describes each model and determines which model yields overestimated results.

2.3 Error Propagation Method

The error propagation technique determines the standard deviation in each body frame direction. To compare the first order and second order method only the xy-body frame's covariance is calculated. Both models require the aircraft body model Eq.2.2 and each variable errors: σ_s, σ_{x0}, σ_{y0}, σ_{z0}, σ_ϕ, and σ_θ.

2.3.1 First Order Model

The general formula for the first order propagation of error [26] is Eq.2.3.

$$\sigma_F^2 = \left(\frac{\partial F\left(x_1,x_2,x_3,\ldots\right)}{\partial x_1}\right)^2 \sigma_{x_1}^2 + \left(\frac{\partial F\left(x_1,x_2,x_3,\ldots\right)}{\partial x_2}\right)^2 \sigma_{x_2}^2 + \left(\frac{\partial F\left(x_1,x_2,x_3,\ldots\right)}{\partial x_3}\right)^2 \sigma_{x_3}^2 + \ldots. \quad (2.3)$$

Using Eq.2.2 and nominal conditions for θ_{body}, the first order error propagation model is

$$\begin{aligned}\sigma_{x(t)} &= \sqrt{\sigma_{x_0}^2 + t^2\left(s^2\sin^2\left(\theta_{body}\right)\sigma_\theta^2 + \cos^2\left(\theta_{body}\right)\sigma_s^2\right)} = \sqrt{\sigma_{x_0}^2 + t^2\sigma_s^2} \\ \sigma_{y(t)} &= \sqrt{\sigma_{y_0}^2 + t^2\left(s^2\cos^2\left(\theta_{body}\right)\sigma_\theta^2 + sin^2\left(\theta_{body}\right)\sigma_s^2\right)} = \sqrt{\sigma_{y_0}^2 + t^2 s^2\sigma_\theta^2}\end{aligned} \tag{2.4}$$

2.3.2 Second Order Model

The general form for the second order error propagation method [26] is Eq.2.5. The second order model includes the covariance between each variable and the partial derivatives between each variable.

$$\sigma_F{}^2 = \bar{g}_0^T\Sigma_x\bar{g}_0 + \frac{1}{2}trace\left(H_0\Sigma_x H_0\Sigma_x\right) \tag{2.5}$$

Applying this to the body frame equation yields

$$\begin{aligned}\bar{g}_0 &= \begin{bmatrix} 1 \\ t sin\left(\theta_{body}\right) \\ st\cos\left(\theta_{body}\right)\end{bmatrix}, H_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & t\cos\left(\theta_{body}\right) \\ 0 & t\cos\left(\theta_{body}\right) & -st\sin\left(\theta_{body}\right)\end{bmatrix}, \\ \Sigma_x &= \begin{bmatrix} \sigma_{x_0}^2 & 0 & 0 \\ 0 & \sigma_s^2 & 0 \\ 0 & 0 & \sigma_\theta^2\end{bmatrix}\end{aligned} \tag{2.6}$$

The second order model assumes independence between all the errors and uses the same first order model assumptions.

In the x-direction:

$$\bar{g}_0 = \begin{bmatrix} 1 \\ t\cos\left(\theta_{body}\right) \\ -st\sin\left(\theta_{body}\right)\end{bmatrix}, H_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t\sin\left(\theta_{body}\right) \\ 0 & -t\sin\left(\theta_{body}\right) & -st\cos\left(\theta_{body}\right)\end{bmatrix}, \Sigma_x = \begin{bmatrix} \sigma_{x_0}^2 & 0 & 0 \\ 0 & \sigma_s^2 & 0 \\ 0 & 0 & \sigma_\theta^2\end{bmatrix} \tag{2.7}$$

Thus σ_x is

$$\begin{aligned}\sigma_x &= \sqrt{\frac{\sigma_\theta^4 s^2 t^2\cos^2\left(\theta_{body}\right)}{2} + \frac{\sigma_\theta^2\sigma_s^2 t^2\sin^2\left(\theta_{body}\right)}{2} + \sigma_{x_0}^2 + \sigma_s^2 t^2\cos^2\left(\theta_{body}\right) + s^2\sigma_\theta^2 t^2\sin^2\left(\theta_{body}\right)} \\ &= \sqrt{\frac{\sigma_\theta^4 s^2 t^2}{2} + \sigma_{x_0}^2 + \sigma_s^2 t^2}\end{aligned} \tag{2.8}$$

In the y-direction:

$$\bar{g}_0 = \begin{bmatrix} 1 \\ t sin\left(\theta_{body}\right) \\ st\cos\left(\theta_{body}\right) \end{bmatrix}, H_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & t\cos\left(\theta_{body}\right) \\ 0 & t\cos\left(\theta_{body}\right) & -st\sin\left(\theta_{body}\right) \end{bmatrix}, \Sigma_x = \begin{bmatrix} \sigma_{x_0}^2 & 0 & 0 \\ 0 & \sigma_s^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{2.9}$$

Thus σ_y is

$$\begin{aligned} \sigma_y &= \sqrt{\frac{\sigma_\theta^4 s^2 t^2 \sin^2\left(\theta_{body}\right)}{2} + \frac{\sigma_\theta^2 \sigma_s^2 t^2 \cos^2\left(\theta_{body}\right)}{2} + \sigma_{y_0}^2 + \sigma_s^2 t^2 \sin^2\left(\theta_{body}\right) + s^2 \sigma_\theta^2 t^2 \cos^2\left(\theta_{body}\right)} \\ &= \sqrt{\frac{\sigma_\theta^2 \sigma_s^2 t^2}{2} + \sigma_{y_0}^2 + s^2 \sigma_\theta^2 t^2} \end{aligned} \tag{2.10}$$

2.4 Comparing First and Second Error Propagation Method

The method with the larger standard deviation in both the x-direction and y-direction is the conservative method. If the difference is negative, then the second order method is conservative. Since σ_{x0} and σ_{y0} are the same,σ_θ, s and σ_s will be varied. For these scenarios σ_{x_0} and σ_{y_0} is 3m. This results in six different scenarios (Table 2.1). Scenarios 1 and 2 are typical scenarios because s is normally the largest variable.

Scenario #	Description
1	$s > \sigma_s > \sigma_\theta$
2	$s > \sigma_\theta > \sigma_s$
3	$\sigma_s > \sigma_\theta > s$
4	$\sigma_s > s > \sigma_\theta$
5	$\sigma_\theta > s > \sigma_s$
6	$\sigma_\theta > \sigma_s > s$

Table 2.1: Comparing error propagation method scenario description

2.4.1 Comparing error propagation methods scenario 1

The scenario 1 parameters are $\sigma_\theta = 2^\circ$, $s = 50\frac{\text{m}}{\text{s}}$ and $\sigma_s = 5\frac{\text{m}}{\text{s}}$. Figure 2.5 shows both methods yield very similar σs, however, the difference is always negative, where both σ's max difference is 0. Thus, the second order method is more conservative.

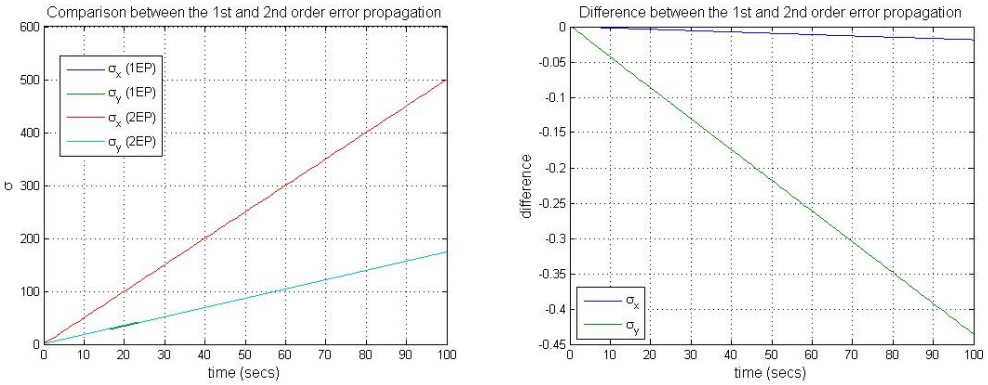

Figure 2.5: Comparing error propagation methods scenario 1, σ values (left) and difference (right)

2.4.2 Comparing error propagation methods scenario 2

The scenario 2 parameters are $\sigma_\theta = 10^\circ$, $s = 5\frac{\text{m}}{\text{s}}$ and $\sigma_s = .15\frac{\text{m}}{\text{s}}$. Figure 2.6 shows both methods yield very similar results for σ_y but the second order method yields a larger σ_x. Since both σ's max difference is 0, the second order method is more conservative.

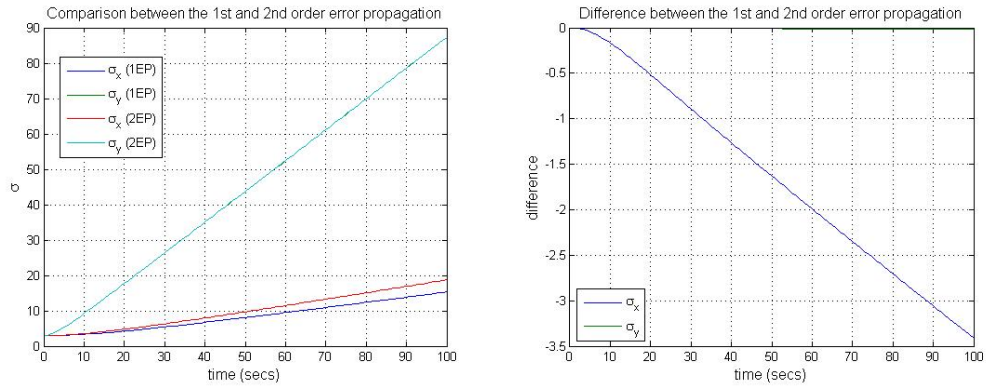

Figure 2.6: Comparing error propagation methods scenario 2, σ values (left) and difference (right)

2.4.3 *Comparing error propagation methods scenario 3*

The scenario 3 parameters are $\sigma_\theta = 40°$, $s = .6\frac{\text{m}}{\text{s}}$ and $\sigma_s = .75\frac{\text{m}}{\text{s}}$. Figure 2.7 shows the second order method is larger for both σs, where the max difference is 0. Thus, the second order method is more conservative.

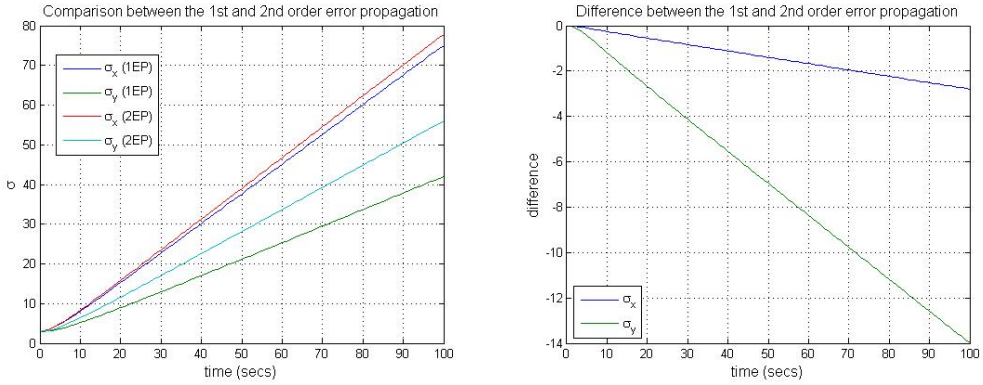

Figure 2.7: Comparing error propagation methods scenario 3, σ values (left) and difference (right)

2.4.4 *Comparing error propagation methods scenario 4*

The scenario 4 parameters are $\sigma_\theta = 20°$, $s = .45\frac{\text{m}}{\text{s}}$ and $\sigma_s = .5\frac{\text{m}}{\text{s}}$. Figure 2.7 shows both methods yield very similar results for σ_x but the second order method yields a larger σ_y. Since both σ's max difference is 0, the second order method is more conservative.

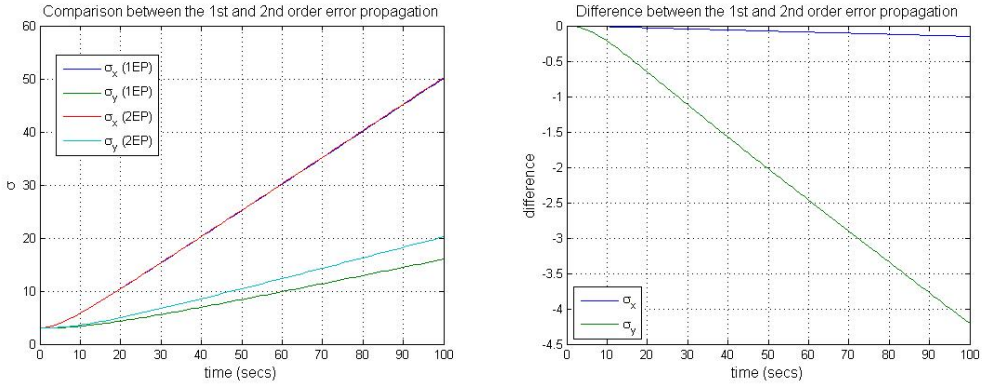

Figure 2.8: Comparing error propagation methods scenario 4 σ values (left) and difference (right)

2.4.5 *Comparing error propagation methods scenario 5*

The scenario 5 parameters are $\sigma_\theta = 40°$, $s = .5\frac{\text{m}}{\text{s}}$ and $\sigma_s = .15\frac{\text{m}}{\text{s}}$. Figure 2.9 shows both methods yield very similar results for σ_y but the second order method yields a larger σ_x. However since both σ's max difference is 0, the second order method is more conservative.

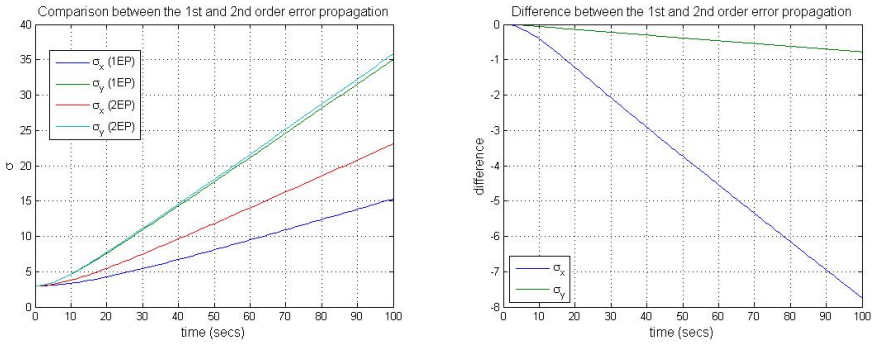

Figure 2.9: Comparing error propagation methods scenario 5, σ values (left) and difference (right)

2.4.6 *Comparing error propagation methods scenario 6*

The scenario 6 parameters are $\sigma_\theta = 40°$, $s = .2\frac{\text{m}}{\text{s}}$ and $\sigma_s = .3\frac{\text{m}}{\text{s}}$. Figure 2.10 shows both methods yield very similar results for both σs where the max difference for both is 0. Thus, the second order method is more conservative.

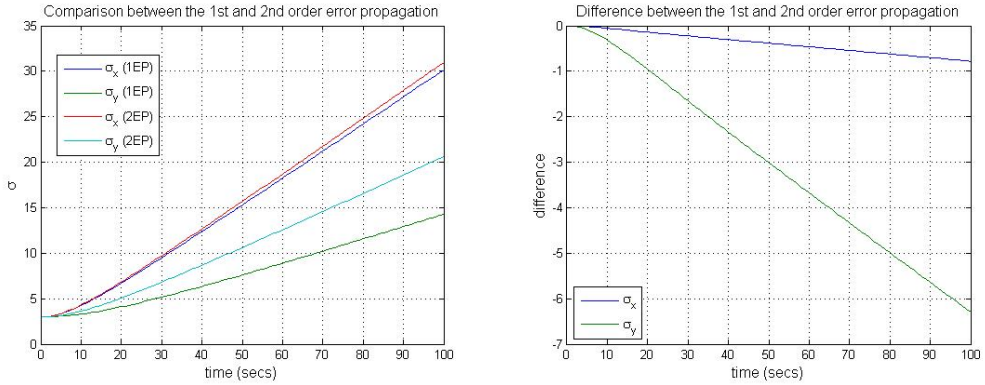

Figure 2.10: Comparing error propagation methods scenario 6, σ values (left) and difference (right)

2.4.7 Calculating σ_{body}

Although one example from each scenario is tested, the second order error propagation model will always conservatively calculate σ because there is an additional nonnegative term. Thus, the standard deviation for each body frame direction is

$$\begin{aligned}\sigma_x &= \sqrt{\frac{\sigma_\theta^4 s^2 t^2}{2} + \sigma_{x_0}^2 + \sigma_s^2 t^2} \\ \sigma_y &= \sqrt{\frac{\sigma_\theta^2 \sigma_s^2 t^2}{2} + \sigma_{y_0}^2 + s^2 \sigma_\theta^2 t^2} \\ \sigma_z &= \sqrt{\frac{\sigma_\phi^4 s^2 t^2}{2} + \sigma_{z_0}^2 + \sigma_s^2 t^2}\end{aligned} \tag{2.11}$$

2.5 Covariance Calculation

The body frame's variance in each direction is the standard deviation squared. Since no rotation is required between the body and Cartesian system the variance in the z-direction is the same.

$$\sigma_z^2(t) = \sigma_{zBody}^2(t) \tag{2.12}$$

Assuming $\sigma_{x,Body}$ and $\sigma_{y,Body}$ are independent the covariance is

$$Cov_{XY\,Body}(t) = \begin{bmatrix} \sigma_{xBody}^2(t) & 0 \\ 0 & \sigma_{yBody}^2(t) \end{bmatrix} \tag{2.13}$$

A direction cosine matrix (DCM) using θ transforms the covariance from the body frame to the Cartesian system. Figure 2.11 illustrates the relationship between the body covariance and the xy-covariance. The Cartesian frame's xy-covariance is determined by Eq.2.14.

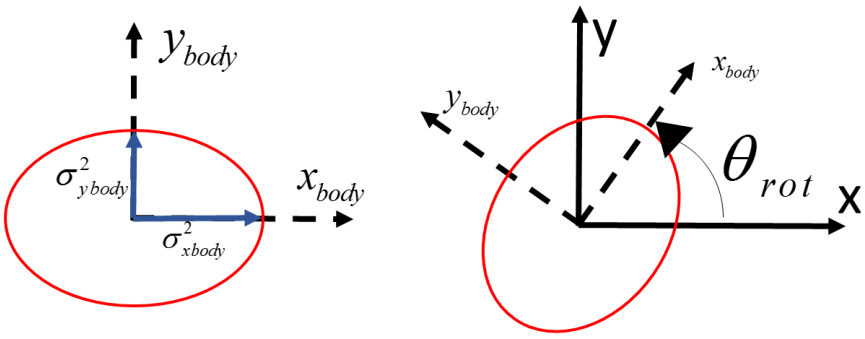

Figure 2.11: Body covariance and relationship between body and Cartesian frame

Assuming independence between σ_{xbody} and σ_{ybody}, the body covariance is calculated. Next the body covariance is transformed into the Cartesian frame using a DCM with rotation angle θ_{rot}.

$$Cov_{xy} = DCM^{-1}_{Body/xy} Cov_{Body} DCM_{Body/xy} = DCM^{T}_{Body/xy} Cov_{Body} DCM_{Body/xy} \tag{2.14}$$

2.6 FSE Free Flight Simulation

The FSE free flight is demonstrated with 3 different scenarios. Scenario 1 is an aircraft ascending with $\sigma_{xbody} > \sigma_{ybody}$, scenario 2 is an aircraft descending with $\sigma_{xbody} < \sigma_{ybody}$, and scenario 3 is an aircraft with $\phi = 0°$. The parameters for all three scenarios are listed in table 2.2.

Scenario #	θ	σ_θ	ϕ	σ_ϕ	s	σ_s	x_0	y_0	σ_x & σ_y	z_0	σ_z
1	30°	1°	10°	3°	28.3 $\frac{\text{m}}{\text{secs}}$	2 $\frac{\text{m}}{\text{secs}}$	0m	0m	3m	100m	10m
2	30°	8°	−10°	3°	28.3 $\frac{\text{m}}{\text{secs}}$	2 $\frac{\text{m}}{\text{secs}}$	0m	0m	3m	100m	10m
3	30°	1°	0°	3°	28.3 $\frac{\text{m}}{\text{secs}}$	2 $\frac{\text{m}}{\text{secs}}$	0m	0m	3m	100m	10m

Table 2.2: FSE free flight test descriptions

2.6.1 *FSE free flight simulation 1*

Figure 2.13 illustrates both the xy-2D distribution and the altitude distribution. As expected the 2D distribution grows faster along the 30° heading than in the perpendicular direction and the distribution's size increases as the prediction time increases. In the altitude direction, the distribution shifts to the right indicating an increase in altitude and flattens as the prediction time increases.

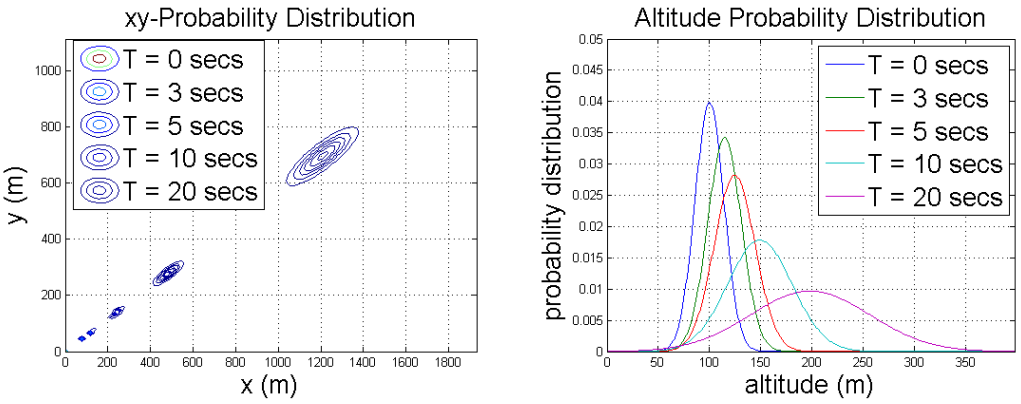

Figure 2.13: FSE free flight simulation 1, xy-distribution (left) and altitude distribution (right)

2.6.2 *FSE free flight simulation 2*

Figure 2.14 illustrates both the xy-2D distribution and the altitude distribution. As expected the 2D distribution grows faster along the 30° angle's perpendicular direction than the 30° angle direction and the distribution's size increases in size as the time horizon increases. In the altitude direction, the distribution shifts to the left indicating a decrease in altitude and flattens as the prediction time increases.

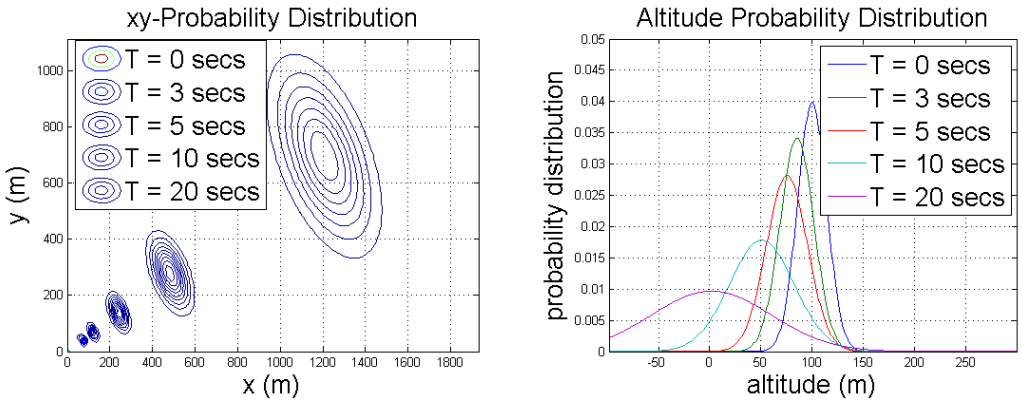

Figure 2.14: FSE free flight simulation 2, xy-distribution (left) and altitude distribution (right)

2.6.3 FSE free flight simulation 3

Figure 2.15 illustrates both the xy-2D distribution and the altitude distribution. Although the 2D distribution is the same as in scenario 1, this scenario examines how the FSE free flight handles the $\phi = 0^\circ$ case. As expected the altitude distribution still flattens because the FSE free flight accounts for altitude changes occurring in the future. This implies as the prediction time increases the 95% confidence altitude range will increase as well.

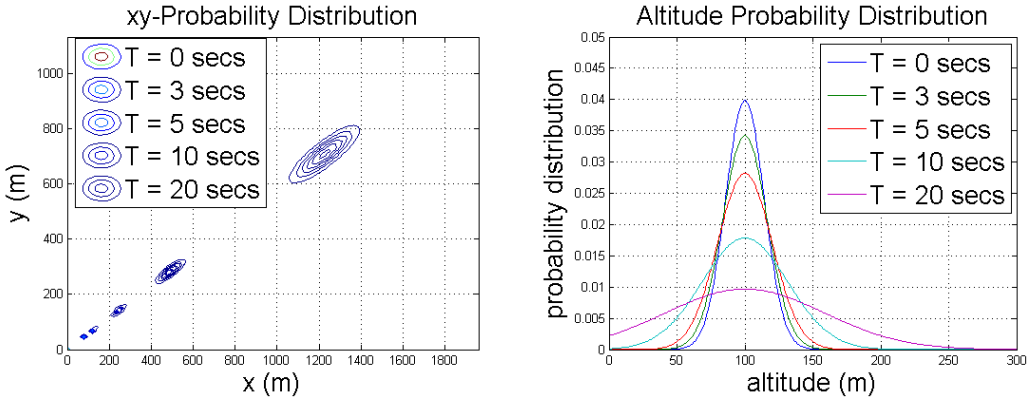

Figure 2.15: FSE free flight simulation 3, xy-distribution (left) and altitude distribution (right)

Chapter 3

FORWARD STATE ESTIMATOR ORBIT

3.1 Introduction

The FSE orbit conservatively predicts an aircraft's future position for a circular orbiting aircraft. An orbiting aircraft flies in a circular pattern exclusively clockwise or counterclockwise at a constant altitude and speed. In addition to FSE free flight inputs, this algorithm requires inputs on the orbit's center, radius and direction. Figure 3.1 illustrates the three different orbit parameters. For this paper the orbit center is defined as

$$orbit_{center} = \begin{bmatrix} x_{orbit} \\ y_{orbit} \\ z_{orbit} \end{bmatrix} \tag{3.1}$$

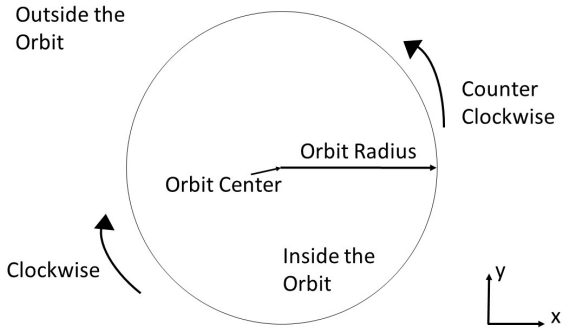

Figure 3.1: Orbit parameter description

An orbit is defined by the oribt center, orbit radius and the direction of travel. An aircraft inside the orbit has an xy-distance from the orbit center less than the orbit radius. And an aircraft outside the orbit has an xy-distance from the orbit center greater than the orbit radius.

3.2 Mean Calculation

The mean calculation is separated into two cases: an aircraft starting on the orbit and an aircraft that is not. Each case is determined by first calculating $dist_{xy}$ and alt_{dif}.

$$dist_{xy} = dist_{xy,Orbit} - orbit_{radius} \tag{3.2}$$

where $dist_{xy,Orbit}$ is the distance in the xy-plane between the orbit center and the aircraft's current position.

$$alt_{dif} = z_{orbit} - z_0 \tag{3.3}$$

If $dist_{xy} = 0$ and $alt_{dif} = 0$, then the aircraft is on the orbit. If not, then the aircraft is not on the orbit (this case is discussed in the next section).

θ_0 is an angle defined the same way polar coordinate angles are.

$$\theta_0 = \text{atan2}\left(\left(y_0 - y_{orbit}\right), \left(x_0 - x_{orbit}\right)\right) \tag{3.4}$$

3.2.1 Aircraft on the orbit

θ_{trav} is the angle the aircraft traveled on the orbit.

$$\theta_{trav} = \frac{st}{orbit_{radius}} \tag{3.5}$$

The aircraft's predicted position depends on the orbit direction traveled.

$$\begin{aligned} &\text{Counterclockwise} : \theta(t) = \theta_0 + \theta_{trav} \\ &\text{Clockwise} : \theta(t) = \theta_0 - \theta_{trav} \end{aligned} \tag{3.6}$$

Finally the angular position is transformed to the mean position.

$$\begin{aligned} x(t) &= x_{orbit} + \cos\left(\theta(t)\right) orbit_{radius} \\ y(t) &= y_{orbit} + \sin\left(\theta(t)\right) orbit_{radius} \\ z(t) &= z_{orbit} \end{aligned} \tag{3.7}$$

3.2.2 Aircraft not on the orbit

There are three different situations when the aircraft is not on the orbit: on the orbit in the xy-plane but not at orbit altitude, outside the orbit and inside the orbit. When $dist_{xy} < 0$, the aircraft is inside an orbit and is outside when $dist_{xy} > 0$.

This algorithm assumes an aircraft not on the orbit proceeds directly to the orbit following a radial path, which differs from Insitu's standard procedure. Normally the aircraft enters the orbit tangentially and not radially [27]. Another assumption is an aircraft not at the orbit altitude flies at a gradual climb angle (ϕ), not to exceed a max climb angle[1], to reach the orbit altitude. This assumption also does not reflect Insitu's standard procedure because the aircraft always flies at the max climb or descend angle until the orbit altitude is reached [28].

Since there is a specified orbit altitude, ϕ has to be carefully defined. The case when $dist_{xy} = 0$ is discussed in the next section.

$$\phi_0 = \left|\text{atan}\left(\frac{alt_{dif}}{dist_{xy}}\right)\right| \Rightarrow \begin{cases} \text{if } alt_{dif} < 0,\ \phi_{initial} = -\phi_0 \\ \text{if } alt_{dif} > 0,\ \phi_{initial} = \phi_0 \end{cases} \tag{3.8}$$

Applying the max climb angle assumption, if $|\phi_{initial}| \leq 10^\circ$, then $\phi = \phi_{initial}$. The other case is if $|\phi_{initial}| > 10^\circ$, then $\phi = \pm 10^\circ$, where ϕ's sign is the same as $\phi_{initial}$'s sign.

t_{alt} is the time the aircraft reaches the correct orbit altitude and t_{orb} is the time the aircraft reaches the orbit in the xy-plane.

$$t_{alt} = \frac{alt_{dif}}{s\sin(\phi)} \tag{3.9}$$

$$t_{orb} = \frac{|dist_{xy}|}{s\cos(\phi)} \tag{3.10}$$

Situation 1: Aircraft on orbit in xy projection but different altitude

There are two different cases for this situation: an aircraft above or below the orbit altitude.

---

[1] This paper assumes the max angle is 10°.

When the aircraft is above the orbit altitude, the aircraft follows the orbit in the xy-plane and $\phi = -10^{\circ}$. For the other case the aircraft performs the same action except $\phi = 10^{\circ}$. Also there are two distinct time periods, $t < t_{alt}$ and $t > t_{alt}$. When $t < t_{alt}$, the z-position is

$$z(t) = z_0 + st\sin(\phi) \tag{3.11}$$

The xy-position is calculated using θ_{trans}, the angle the aircraft travels in the orbit while reaching the orbit altitude.

$$\theta_{trans} = \frac{s|\cos(\phi)|t}{orbit_{radius}} \tag{3.12}$$

θ is found by substituting θ_{trans} for θ_{trav} in Eq.3.6, then $x(t)$ and $y(t)$ is calculated using Eq.3.7. When $t > t_{alt}$, $\theta_{transNotAlt}$, the angle the aircraft traveled at t_{alt}, is found by substituting t_{alt} for t in Eq.3.12. Then $\theta_{transOnOrb}$, the angle the aircraft traveled at $t - t_{alt}$, is determined by substituting $t - t_{alt}$ for t in Eq.3.5. $\theta(t)$ is calculated by using Eq.3.6 and Eq.3.13. Then, the mean position is found by evaluating Eq.3.7 and applying $\theta(t)$.

$$\theta_{trav} = \theta_{transNotAlt} + \theta_{transOnOrb} \tag{3.13}$$

Situation 2: Aircraft is outside the orbit

Although there are three different cases for this situation: aircraft is at a higher, lower or at the orbit altitude, the logic for each case is the same. For each of these cases they all have the same three distinct time periods: $t < t_{orb}$, $t < t_{alt}$ and $t > t_{alt}$. When $t < t_{orb}$, the aircraft heads directly to the orbit and enters the orbit radially the aircraft follows the angle ψ.

$$\psi = \theta_0 + 180^{\circ} \tag{3.14}$$

Then the mean is calculated using Eq.2.1 and substituting ψ for θ. When $t < t_{alt}$, the aircraft is on the orbit in the xy plane, which is similar to situation 1. θ_{trans} is calculated by replacing t with $t - t_{orb}$ into Eq.3.12. Then the mean is calculated using Eq.3.7 and Eq.3.11. When $t > t_{alt}$, situation 1's algorithm during the same time period is applied, except $t_{alt} - t_{orb}$ is substituted for $t - t_{alt}$ in Eq.3.12.

Situation 3: Aircraft is inside the orbit

This situation follows the same logic as situation 2 except substitute ψ_{inside} for ψ.

$$\psi_{inside} = \theta_{initial} \tag{3.15}$$

3.3 Covariance Calculation

The FSE orbit's covariance calculation is similar to the FSE free flight covariance calculation, except reasonable limits are applied to each body frame direction since the aircraft's intent is known. The body frame directions for the FSE orbit is altitude (body frame's +z-axis), orbit's radial direction (body frame's +y-axis), and orbit's tangential direction (body frame's +x-axis). The xy-limits are applied once the aircraft is on the orbit in the xy-plane and the altitude limits are applied once the aircraft reaches the orbit altitude. A reasonable assumption is the error in the tangential direction (σ_{xError}) cannot exceed the orbit's diameter. In this paper $\sigma_{yError} = 5$m and $\sigma_{zError} = 20$m.[2]

t_{Error} is the prediction time when $\sigma = \sigma_{Errror}$ and is calculated by Eq.3.16. t_{Error} is a real solution when $\sigma_{xError} > \sigma_{x0}$, $\sigma_{yError} > \sigma_{y0}$, and $\sigma_{zError} > \sigma_{z0}$. These are reasonable conditions because if the GPS error is larger than the max error, then an aircraft cannot be guaranteed to be within a max error. By calculating t_{Error}, the FSE orbit applies the max error when $t \geq t_{Error}$. Furthermore, t_{xError} and t_{yError} are applied when the aircraft is on the orbit's xy-plane and t_{zError} is applied when the aircraft is at the orbit altitude. For the cases when $t_{orb} > t_{xError}$, $t_{orb} > t_{yError}$, and $t_{alt} > t_{zError}$, the error grows normally until t_{orb} or t_{alt} is reached. Once $t > t_{orb}$ or $t > t_{alt}$, then the max errors are applied. This scenario is further discussed in limiting σ scenario 3.

$$\begin{aligned} t_{xError} &= \frac{\sigma_{xError}^2 - \sigma_{x_0}^2}{\frac{\sigma_\theta^4 s^2}{2} + \sigma_s^2} \\ t_{yError} &= \frac{\sigma_{yError}^2 - \sigma_{y_0}^2}{\frac{\sigma_\theta^2 \sigma_s^2}{2} + s^2 \sigma_\theta^2} \\ t_{zError} &= \frac{\sigma_{zError}^2 - \sigma_{z_0}^2}{\frac{\sigma_\phi^4 s^2}{2} + \sigma_s^2} \end{aligned} \tag{3.16}$$

Also the FSE orbit's DCM rotation angle depends on the orbit direction and whether an aircraft is outside or inside the orbit. If the aircraft is not on the orbit in the xy-plane, ψ and ψ_{inside} is applied

[2]These max error values can be operator specified.

appropriately. The DCM's rotation angle on the orbit in the xy-plane is defined as

$$\begin{aligned} \text{Counterclockwise: } \theta_{rot} &= \theta + \frac{\pi}{2} \\ \text{Clockwise: } \theta_{rot} &= \theta - \frac{\pi}{2} \end{aligned} \tag{3.17}$$

3.4 Matlab Simulation

There are two simulations: limiting σ and predicting the aircraft's mean position.

3.4.1 Limiting σ

This demonstration has three examples with the same σ limits, which are $\sigma_{yError} = 5$m, $\sigma_{xError} = 200$m, $orbit_{radius} = 100$m and $\sigma_{zError} = 20$m. Table 3.1 describes each scenario.

Scenario #	σ_θ	s	σ_s	σ_ϕ	t_{orbit}	$t_{altitude}$
1	2°	50 $\frac{\text{m}}{\text{sec}}$	5 $\frac{\text{m}}{\text{sec}}$	10°	0 seconds	0 seconds
2	20°	50 $\frac{\text{m}}{\text{sec}}$	3 $\frac{\text{m}}{\text{sec}}$	5°	0 seconds	0 seconds
3	2°	50 $\frac{\text{m}}{\text{sec}}$	5 $\frac{\text{m}}{\text{sec}}$	10°	12.1 seconds	19.8 seconds

Table 3.1: Limiting σ test description

Limiting σ scenario 1

The scenario 1 parameters results in $\sigma_x \geq \sigma_y$, $t_{xError} = 40$ seconds, $t_{yError} = 2.3$ seconds, and $t_{zError} = 3.4$ seconds and σ is illustrated in Figure 3.3. As expected σ grows when $t < t_{Error}$ and is constant when $t > t_{Error}$.

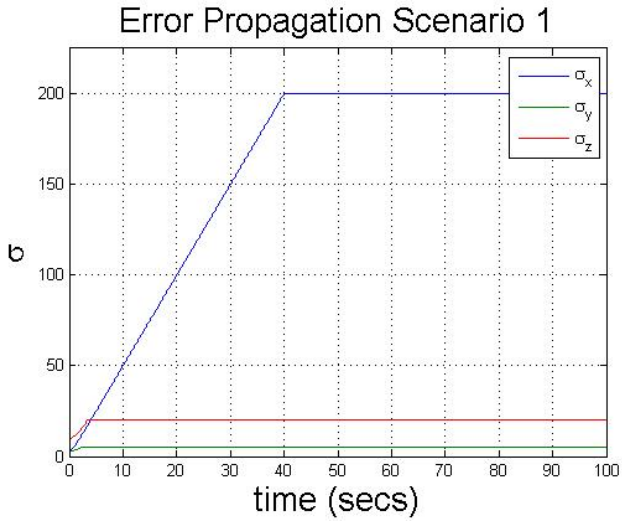

Figure 3.3: Limiting σ scenario 1, σ growth in radial, altitude and tangential direction

Limiting σ scenario 2

Scenario 2 has $\sigma_x \leq \sigma_y$ with $t_{xError} = 38.1$ seconds, $t_{yError} = .2$ seconds, and $t_{zError} = 5.8$ seconds and σ is illustrated in Figure 3.4. As expected σ grows when $t < t_{Error}$ and is constant when $t > t_{Error}$. Since $t_{yError} << t_{xError}$, σ_y is held constant at σ_{yError}.

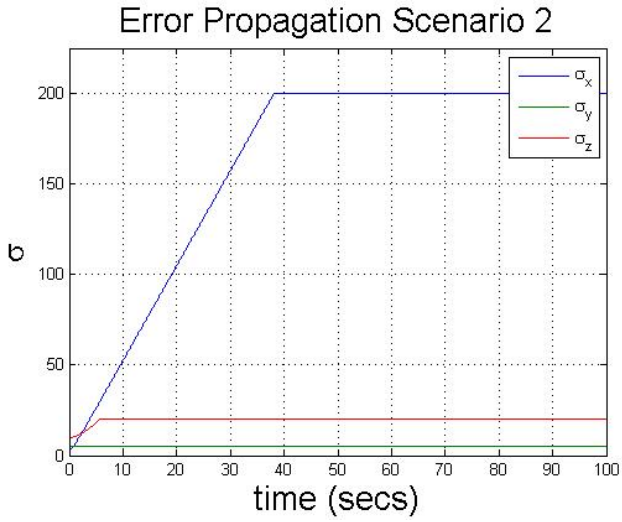

Figure 3.4: Limiting σ scenario 2, σ growth in radial, altitude and tangential direction

Limiting σ scenario 3

Scenario 3's parameters are the same as scenario 1's except $\phi = 0°$. This scenario demonstrates how σ can be larger than σ_{Error}. Figure 3.5 illustrates scenario 3's results. As expected σ grows when $t < t_{orbit}$ and $t < t_{alt}$, however when $t > t_{orbit}$ and $t > t_{alt}$, σ is limited by σ_{Error}. Also since $t_{orbit} < t_{altitude}$ σ_z grows longer than σ_y.

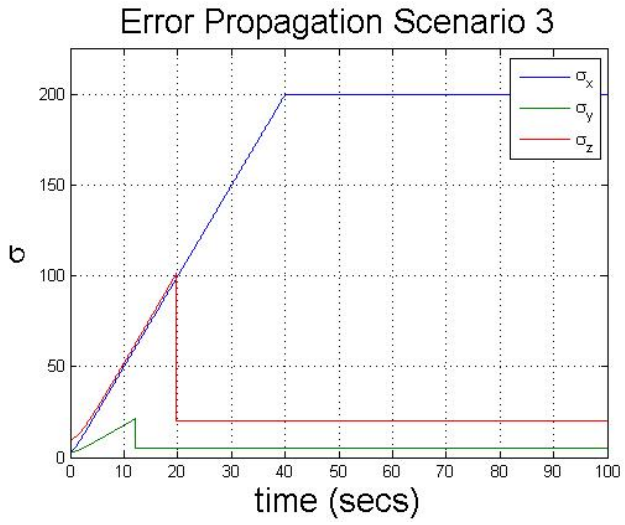

Figure 3.5: Limiting σ scenario 3, σ growth in radial, altitude and tangential direction

3.4.2 Predicting Future Aircraft's Mean

This simulation demonstrates how the FSE orbit handles different initial aircraft position. For all 13 scenarios, the orbit radius is 100m and the orbit center is $orbit_{center} = \begin{bmatrix} 0\text{m} & 0\text{m} & 100\text{m} \end{bmatrix}^{\mathrm{T}}$.

In each scenario an *X* represents the aircraft's position. Each figure contains three graphs, which are from left to right: the aircraft's position in the xy-plane, xz-plane and yz-plane. Table 3.2 summarizes each scenario.

Scenario #	Description
1	aircraft on orbit (clockwise)
2	aircraft on orbit in xy-plane at higher altitude (counterclockwise)
3	aircraft on orbit in xy-plane at lower altitude (clockwise)
4	aircraft is outside the orbit at orbit altitude (counterclockwise)
5	aircraft is outside the orbit and above orbit altitude with $-10^\circ < \phi < 0^\circ$ (counterclockwise)
6	aircraft is outside the orbit and below orbit altitude with $0^\circ < \phi < 10^\circ$ (clockwise)
7	aircraft is outside the orbit and above orbit altitude with $\phi < -10^\circ$ (clockwise)
8	aircraft is outside the orbit and below orbit altitude with $\phi > 10^\circ$ (clockwise)
9	aircraft is inside the orbit and at orbit altitude (counterclockwise)
10	aircraft is inside the orbit and above orbit altitude with $\phi < -10^\circ$ (counterclockwise)
11	aircraft is inside the orbit and below the orbit altitude with $\phi > 10^\circ$ (clockwise)
12	aircraft is inside the orbit and below the orbit altitude with $0^\circ < \phi < 10^\circ$ (counterclockwise)
13	aircraft is inside the orbit and above the orbit altitude with $-10^\circ < \phi < 0^\circ$ (clockwise)

Table 3.2: Predicting the aircraft's mean test description

Predicting Aircraft Position Scenario 1

Scenario 1 has an aircraft on orbit traveling clockwise with the initial position and velocity as

$$pos_0 = \begin{bmatrix} 100\text{m} \\ 0\text{m} \\ 100\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} -1\frac{\text{m}}{\text{s}} \\ -1\frac{\text{m}}{\text{s}} \\ 0\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.18}$$

Figure 3.6 illustrates the FSE orbit mean position results for the first 70 seconds. While Figure 3.7 illustrates the FSE orbit mean position results for the first 700 seconds. As expected the aircraft travels in a circle in the xy-plane and does not vary in altitude.

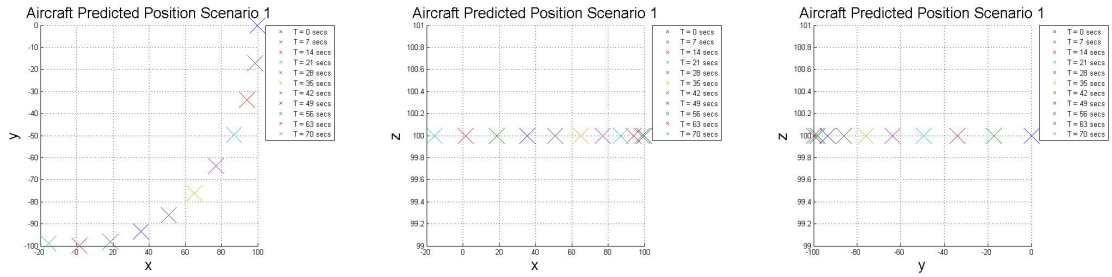

Figure 3.6: Predicting aircraft position scenario 1, first 70 seconds

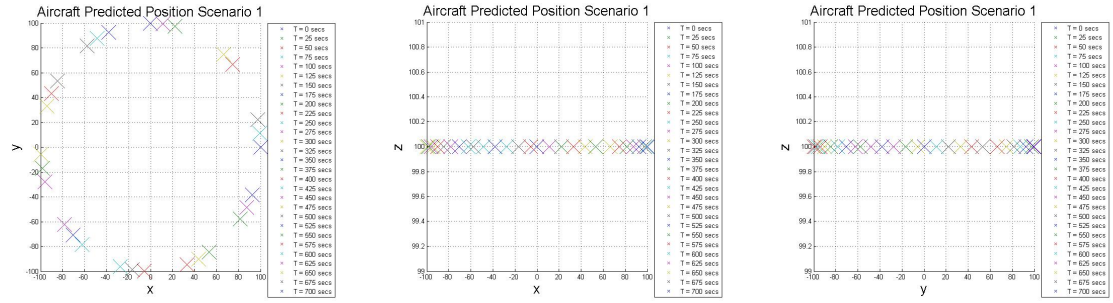

Figure 3.7: Predicting aircraft position scenario 1, first 700 seconds

Predicting Aircraft Position Scenario 2

Scenario 2 has an aircraft on orbit in the xy-plane but above the orbit altitude and traveling counter-clockwise. The initial position and velocity is

$$pos_0 = \begin{bmatrix} 100\text{m} \\ 0\text{m} \\ 150\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} 0\frac{\text{m}}{\text{s}} \\ 4\frac{\text{m}}{\text{s}} \\ -3\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.19}$$

Figure 3.6 illustrates the FSE orbit mean position results for the first 150 seconds. As expected the aircraft orbits counterclockwise in the xy-plane while descending until 57.6 seconds. After this time the aircraft maintains the orbit altitude.

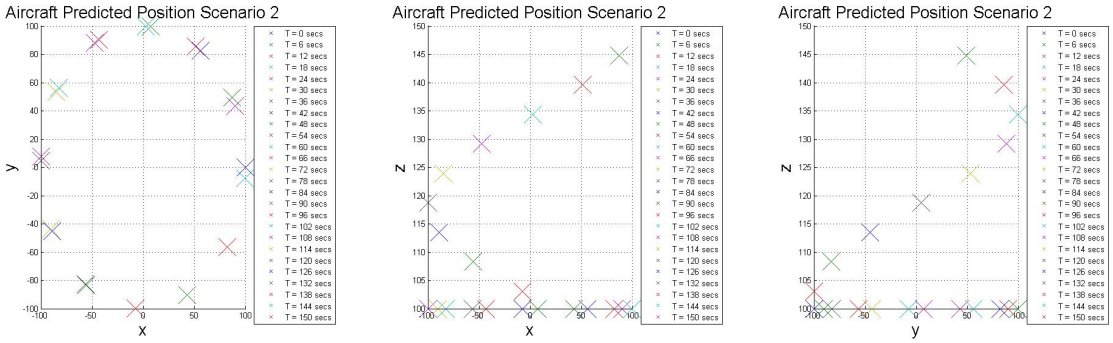

Figure 3.8: Predicting aircraft position scenario 2, first 150 seconds

Predicting Aircraft Position Scenario 3

Scenario 3 has an aircraft on orbit in the xy-plane but below the orbit altitude and traveling clockwise with the initial position and velocity as

$$pos_0 = \begin{bmatrix} 0\text{m} \\ -100\text{m} \\ 50\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} 0\frac{\text{m}}{\text{s}} \\ 4\frac{\text{m}}{\text{s}} \\ 3\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.20}$$

Figure 3.9 illustrates the FSE orbit mean position results for the first 150 seconds. As expected the aircraft orbits clockwise in the xy-plane while ascending until 57.6 seconds. At this time the aircraft reaches the orbit altitude and remains at this altitude.

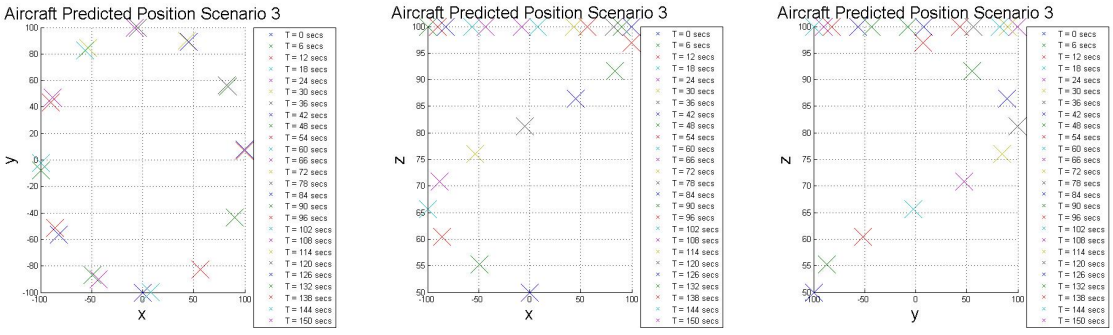

Figure 3.9: Predicting aircraft position scenario 3, first 150 seconds

Predicting Aircraft Position Scenario 4

Scenario 4 has an aircraft outside the orbit but at the orbit altitude traveling counterclockwise. The initial position and velocity is

$$pos_0 = \begin{bmatrix} 0\text{m} \\ -150\text{m} \\ 100\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} 0\frac{\text{m}}{\text{s}} \\ 5\frac{\text{m}}{\text{s}} \\ 0\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.21}$$

Figure 3.10 illustrates the FSE orbit mean position results for the first 150 seconds. As expected the aircraft maintains the altitude and the x-position but heads in the +y-direction until the orbit is reached for the first 10 seconds. At 10 seconds the aircraft is at the orbit and orbits counterclockwise.

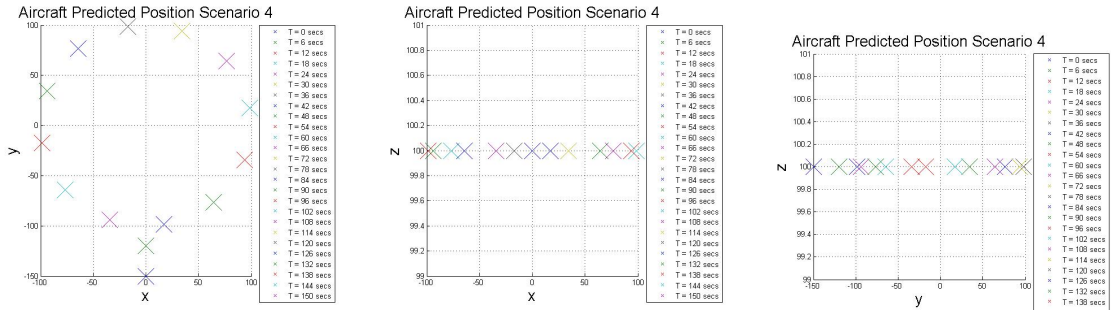

Figure 3.10: Predicting aircraft position scenario 4, first 150 seconds

Predicting Aircraft Position Scenario 5

Scenario 5 has an aircraft outside the orbit but above the orbit altitude and traveling counterclockwise. Also $-10^\circ < \phi < 0^\circ$ because the initial position and velocity is

$$pos_0 = \begin{bmatrix} 0\text{m} \\ -150\text{m} \\ 108\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} 0\frac{\text{m}}{\text{s}} \\ 3\frac{\text{m}}{\text{s}} \\ -4\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.22}$$

Figure 3.11 illustrates the FSE orbit mean position results for the first 100 seconds. As expected the aircraft descends and flies in the +y-direction, while maintaining the x-position for the first 10 seconds. After 10 seconds the aircraft is on the orbit and travels counterclockwise.

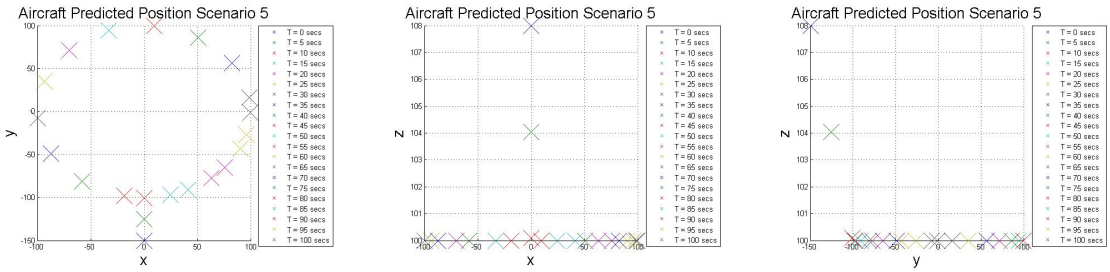

Figure 3.11: Predicting aircraft position scenario 5, first 100 seconds

Predicting Aircraft Position Scenario 6

Scenario 6 has an aircraft outside the orbit and below the orbit altitude traveling clockwise. Due to the initial position and velocity $0^\circ < \phi < 10^\circ$.

$$pos_0 = \begin{bmatrix} 0\text{m} \\ -150\text{m} \\ 92\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} 0\frac{\text{m}}{\text{s}} \\ 3\frac{\text{m}}{\text{s}} \\ 4\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.23}$$

Figure 3.12 illustrates the FSE orbit mean position results for the first 100 seconds. As expected the aircraft ascends and flies in the +y-direction, while maintaining the x-position for the first 10 seconds. After 10 seconds the aircraft reaches the orbit and travels clockwise.

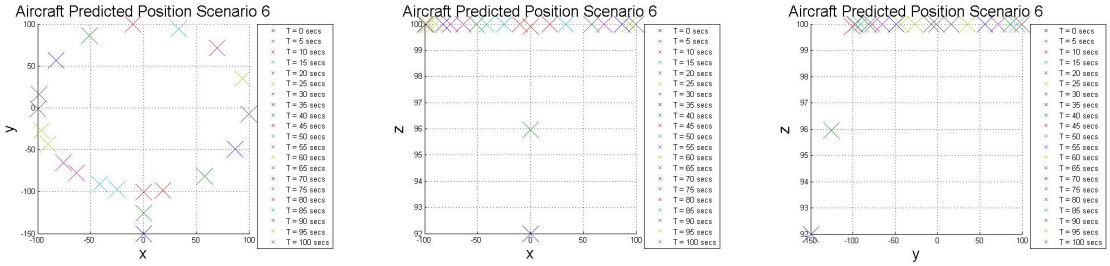

Figure 3.12: Predicting aircraft position scenario 6, first 100 seconds

Predicting Aircraft Position Scenario 7

Scenario 7 has an aircraft outside the orbit but above the orbit altitude traveling clockwise. Also $\phi < -10^\circ$ since the initial position and velocity is

$$pos_0 = \begin{bmatrix} 120\text{m} \\ 30\text{m} \\ 120\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} -1\frac{\text{m}}{\text{s}} \\ -3\frac{\text{m}}{\text{s}} \\ -4\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.24}$$

Figure 3.13 illustrates the FSE orbit mean position results for the first 24 seconds. For this scenario, $t_{orb} = 4.7$ seconds and $t_{alt} = 22.6$ seconds. As expected the aircraft descends and flies in the -x-direction and -y-direction until t_{orb}. At t_{orb} the aircraft is on the orbit in the xy-plane but is still above the orbit. Thus, the aircraft continues to descend and follows the orbit in the xy-plane. During $t_{orb} < t < t_{alt}$, the aircraft's position is similar to scenario 2. At t_{alt}, the aircraft reaches the orbit and continues to travel clockwise as illustrated by figure 3.14.

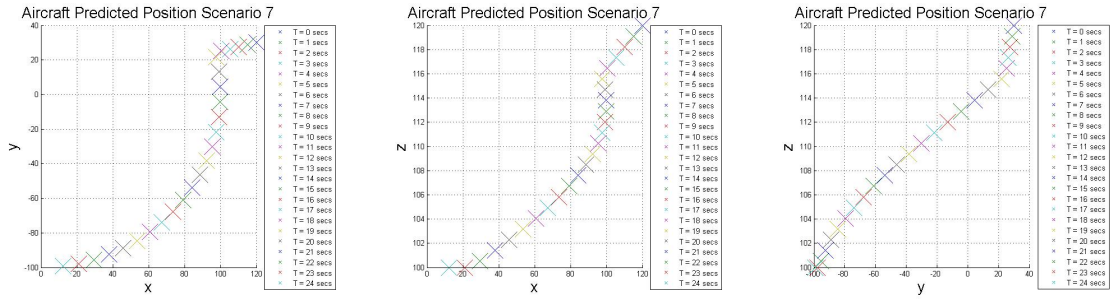

Figure 3.13: Predicting aircraft position scenario 7, first 24 seconds

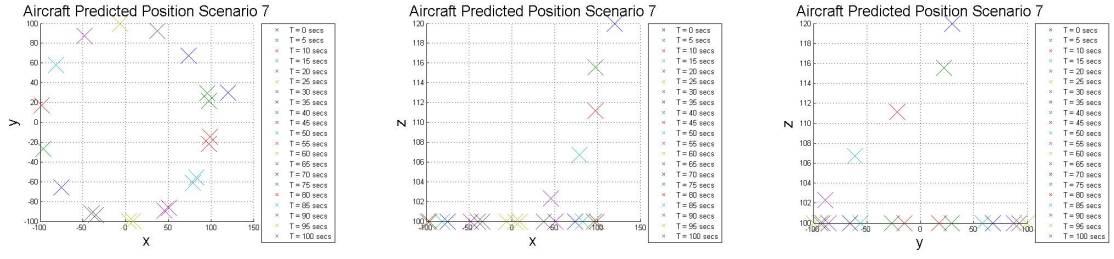

Figure 3.14: Predicting aircraft position scenario 7, first 100 seconds

Predicting Aircraft Position Scenario 8

Scenario 8 has an aircraft outside the orbit and below the orbit altitude traveling clockwise. $\phi > 10^{\circ}$ because the initial position and velocity is

$$pos_0 = \begin{bmatrix} -120\text{m} \\ 120\text{m} \\ 80\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} 3\frac{\text{m}}{\text{s}} \\ -3\frac{\text{m}}{\text{s}} \\ 4\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.25}$$

Figure 3.15 illustrates the FSE orbit mean position results for the first 24 seconds. For this scenario, $t_{orb} = 12.14$ seconds and $t_{alt} = 19.75$ seconds. As expected the aircraft ascends and flies in the +x-direction and -y-direction until t_{orb}. At t_{orb} the aircraft is on the orbit in the xy-plane but is still below the orbit. Thus, the aircraft continues to ascend and follows the orbit in the xy-plane. During $t_{orb} < t < t_{alt}$, the aircraft's position is similar to scenario 3. At t_{alt}, the aircraft reaches the orbit and continues to travel clockwise as illustrated by Figure 3.16.

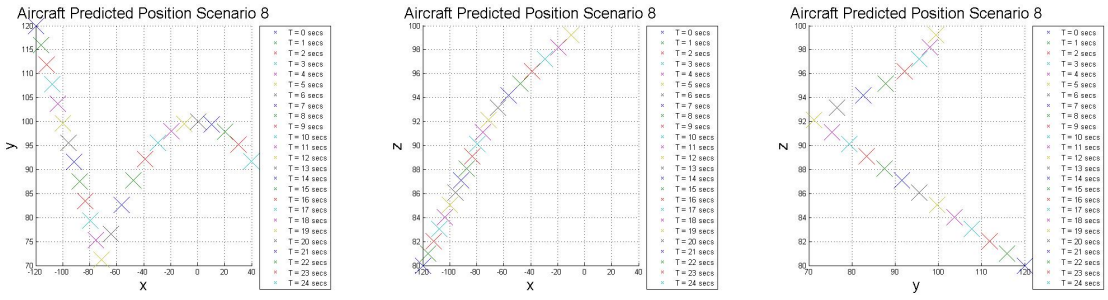

Figure 3.15: Predicting aircraft position scenario 8, first 24 seconds

‘

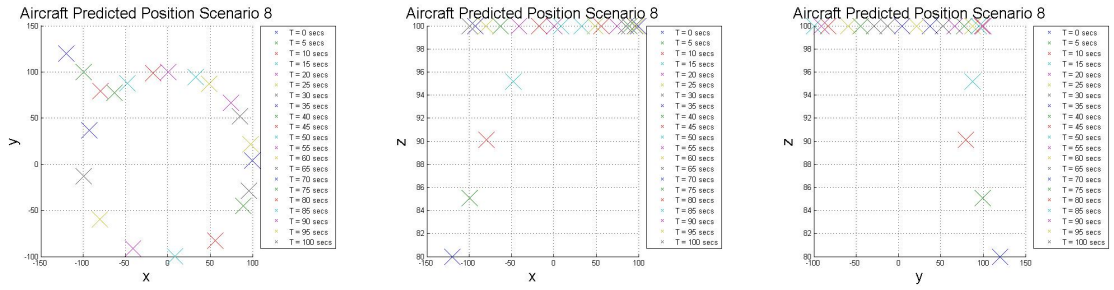

Figure 3.16: Predicting aircraft position scenario 8, first 100 seconds

Predicting Aircraft Position Scenario 9

Scenario 9 has an aircraft inside the orbit at the orbit altitude traveling counterclockwise. Also the initial position and velocity is

$$pos_0 = \begin{bmatrix} 0\text{m} \\ -20\text{m} \\ 100\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} 0\frac{\text{m}}{\text{s}} \\ 20\frac{\text{m}}{\text{s}} \\ 0\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.26}$$

Figure 3.17 illustrates the FSE orbit mean position results for the first 20 seconds. As expected the aircraft maintains altitude and the x-position while flying in the -y-direction until the orbit is reached. Once the orbit is reached, the aircraft stays on the orbit and travels counterclockwise.

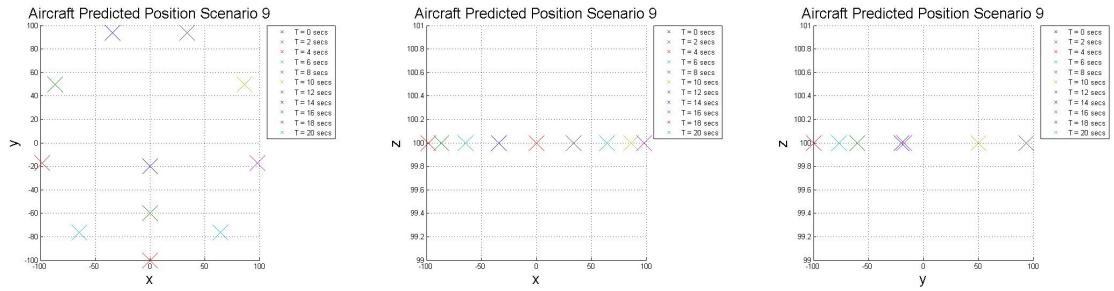

Figure 3.17: Predicting aircraft position scenario 9, first 20 seconds

Predicting Aircraft Position Scenario 10

Scenario 10 has an aircraft inside the orbit and above the orbit altitude traveling counterclockwise. The initial position and velocity results in $\phi < -10^{\circ}$.

$$pos_0 = \begin{bmatrix} 80\text{m} \\ 0\text{m} \\ 105\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} 3\frac{\text{m}}{\text{s}} \\ 0\frac{\text{m}}{\text{s}} \\ -4\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.27}$$

Figure 3.18 illustrates the FSE orbit mean position results for the first 100 seconds. As expected the aircraft descends and maintains the y-position while flying in the +x-direction for $t < t_{orb}$. Then the aircraft continues to descend while following the orbit in the xy-plane when $t_{orb} < t < t_{alt}$. Once the aircraft reaches the orbit, the aircraft orbits counterclockwise.

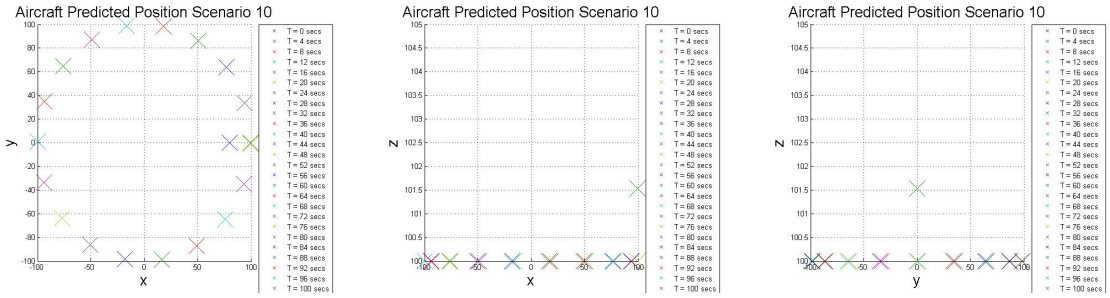

Figure 3.18: Predicting aircraft position scenario 10, first 100 seconds

Predicting Aircraft Position Scenario 11

Scenario 11 has an aircraft inside the orbit and below the orbit altitude traveling clockwise. The initial position and velocity yields a $\phi > 10^{\circ}$.

$$pos_0 = \begin{bmatrix} 98\text{m} \\ 0\text{m} \\ 85\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} 2\frac{\text{m}}{\text{s}} \\ 0\frac{\text{m}}{\text{s}} \\ 4\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.28}$$

Figure 3.19 illustrates the FSE orbit mean position results for the first 100 seconds. As expected the aircraft ascends and maintains the y-position while flying in the +x-direction for $t < t_{orb}$. The

aircraft continues to ascend while following the orbit in the xy-plane during $t_{orb} < t < t_{alt}$. Once the aircraft reaches the orbit, the aircraft orbits counterclockwise.

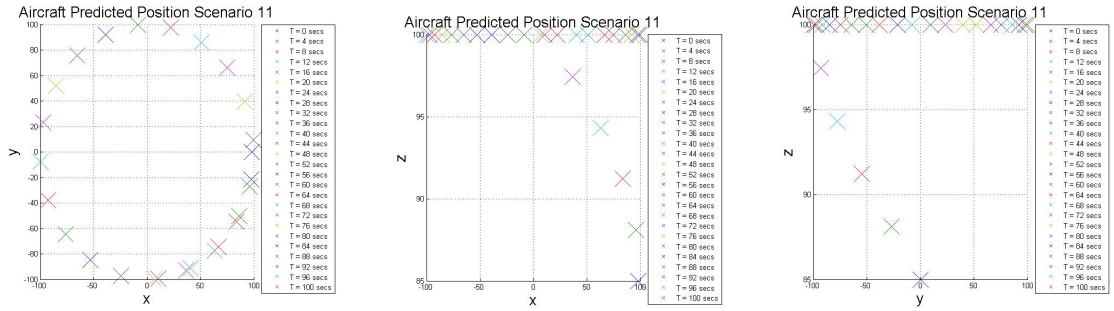

Figure 3.19: Predicting aircraft position scenario 11, first 100 seconds

Predicting Aircraft Position Scenario 12

Scenario 12 has an aircraft inside the orbit but below the orbit altitude traveling counterclockwise. This scenario has $0° < \phi < 10°$ because the initial position and velocity is

$$pos_0 = \begin{bmatrix} 80\text{m} \\ 2\text{m} \\ 98\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} 2\frac{\text{m}}{\text{s}} \\ 3\frac{\text{m}}{\text{s}} \\ 1\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.29}$$

Figure 3.20 illustrates the FSE orbit mean position results for the first 100 seconds. As expected the aircraft ascends in altitude and maintains the x-position while flying in the +y-direction until the orbit is reached. Once the aircraft reaches the orbit, the aircraft orbits counterclockwise.

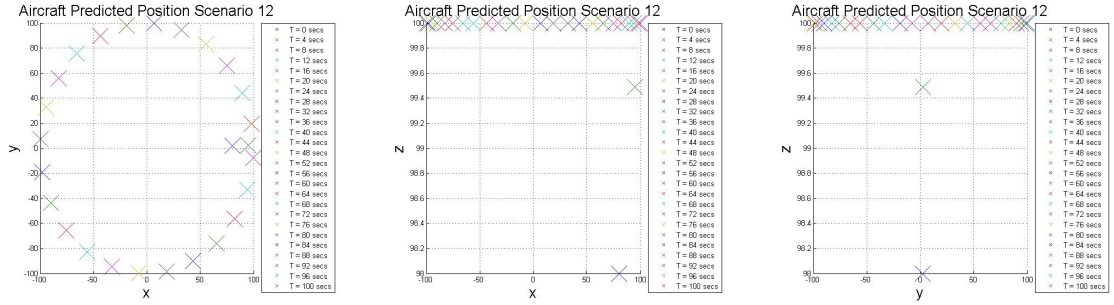

Figure 3.20: Predicting aircraft position scenario 12, first 100 seconds

Predicting Aircraft Position Scenario 13

Scenario 13 has an aircraft inside the orbit and above the orbit altitude traveling clockwise. The initial position and velocity yields a $-10^\circ < \phi < 0^\circ$.

$$pos_0 = \begin{bmatrix} -80\text{m} \\ -2\text{m} \\ 102\text{m} \end{bmatrix}; vel_0 = \begin{bmatrix} -2\frac{\text{m}}{\text{s}} \\ -3\frac{\text{m}}{\text{s}} \\ -1\frac{\text{m}}{\text{s}} \end{bmatrix} \tag{3.30}$$

Figure 3.21 illustrates the FSE orbit mean position results for the first 100 seconds. As expected the aircraft descends in altitude and flies in the -x-direction and -y-direction until the orbit is reached. For $t > t_{orb}$ the aircraft travels clockwise.

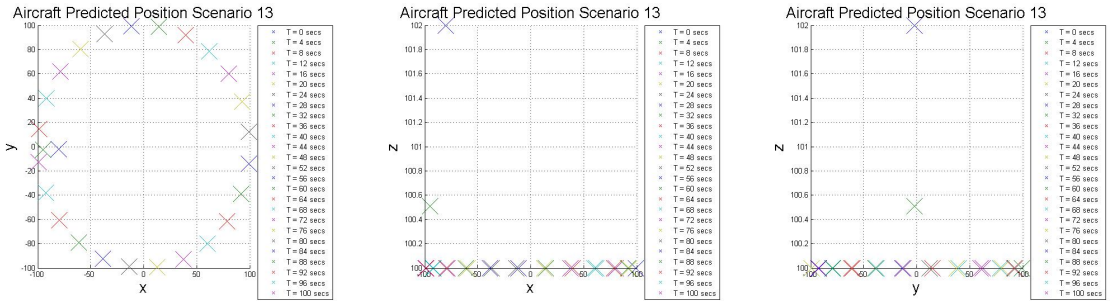

Figure 3.21: Predicting aircraft position scenario 13, first 100 seconds

Chapter 4

AIRSPACE CONFLICT CALCULATOR

4.1 Introduction

The airspace conflict calculator requires an FSE result and an airspace. The airspace is modeled as an extruded non-complex 2D convex or non-convex polygon with vertical side walls. Thus the required inputs to describe the airspace are: a list of each 2D non-complex polygon vertex, airspace's height and the airspace's center altitude. The airspace conflict calculator determines the probability an aircraft will be within an airspace. The FSE result provides both the xy-position's 2D Gaussian distribution's covariance $(\Sigma(t))$ and mean $(\mu(t))$ and the altitude's 1D Gaussian distribution's variance $\left(\sigma_z^2(t)\right)$ and mean $(\bar{z}(t))$. Furthermore in this section an FSE result is depicted as a beige elliptical cylinder, which encompasses the mixed distribution's 95% confidence level.

4.1.1 Gaussian Distributions

A 2D Gaussian distribution function Eq.4.1 [29], defined by the FSE determined covariance $(\Sigma(t))$ and mean $(\mu(t))$, models the aircraft's position in the xy-plane.

$$p_{xy}(t) = \left[(2\pi)^2 \det(\Sigma(t))\right]^{-\frac{1}{2}} e^{-\frac{1}{2}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \mu(t)\right)^{\mathrm{T}} \Sigma(t)^{-1} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \mu(t)\right)} \tag{4.1}$$

where,

$$\begin{aligned} \mu(t) &= \begin{bmatrix} \bar{x}(t) \\ \bar{y}(t) \end{bmatrix} \\ \Sigma(t) &= \begin{bmatrix} \sigma_x^2(t) & \operatorname{cov}(x(t), y(t)) \\ \operatorname{cov}(x(t), y(t)) & \sigma_y^2(t) \end{bmatrix} \end{aligned} \tag{4.2}$$

A 1D Gaussian distribution function Eq.4.3 [29], where the mean and variance is determined by the FSE, represents the aircraft's altitude position.

$$p_z(t) = \frac{e^{\frac{-(z-\bar{z}(t))^2}{2\sigma_z^2(t)}}}{\sqrt{2\pi\sigma_z^2(t)}} \tag{4.3}$$

The probability in the xy-plane ($P_{xy}(t)$) is calculated by integrating a zero off diagonal covariance's 2D Gaussian probability distribution function Eq.4.2. Obtaining a zero off diagonal covariance matrix is discussed in the next section [29]. The probability in the altitude direction ($P_z(t)$) is calculated by Eq.4.5 [29].

$$\begin{aligned} P_{xy}(t) &= \int_{y_0}^{y_f}\int_{x_0}^{x_f}\left[2\pi\det\left(\Sigma(t)\right)\right]^{-\frac{1}{2}} e^{-\frac{1}{2}\left(\begin{bmatrix} x \\ y \end{bmatrix}-\mu(t)\right)^{\mathrm{T}}\Sigma(t)^{-1}\left(\begin{bmatrix} x \\ y \end{bmatrix}-\mu(t)\right)} dxdy \\ &= \frac{1}{4}\left(\mathrm{Erf}\left(\frac{\mathrm{x_f}-\bar{\mathrm{x}}(\mathrm{t})}{\sigma_{\mathrm{x}}(\mathrm{t})\sqrt{2}}\right)-\mathrm{Erf}\left(\frac{\mathrm{x_0}-\bar{\mathrm{x}}(\mathrm{t})}{\sigma_{\mathrm{x}}(\mathrm{t})\sqrt{2}}\right)\right)\left(\mathrm{Erf}\left(\frac{\mathrm{y_f}-\bar{\mathrm{y}}(\mathrm{t})}{\sigma_{\mathrm{y}}(\mathrm{t})\sqrt{2}}\right)-\mathrm{Erf}\left(\frac{\mathrm{y_0}-\bar{\mathrm{y}}(\mathrm{t})}{\sigma_{\mathrm{y}}(\mathrm{t})\sqrt{2}}\right)\right) \end{aligned} \tag{4.4}$$

$$P_z(t) = \int_{z_0}^{z_f} \frac{e^{\frac{-(z-\bar{z}(t))^2}{2\sigma_z^2(t)}}}{\sqrt{2\pi\sigma_z^2(t)}} dz = \frac{1}{2}\left(\mathrm{Erf}\left(\frac{\mathrm{z_f}-\bar{\mathrm{z}}(\mathrm{t})}{\sigma_{\mathrm{z}}(\mathrm{t})\sqrt{2}}\right)-\mathrm{Erf}\left(\frac{\mathrm{z_0}-\bar{\mathrm{z}}(\mathrm{t})}{\sigma_{\mathrm{z}}(\mathrm{t})\sqrt{2}}\right)\right) \tag{4.5}$$

4.2 Airspace Conflict Calculator Steps

Given an FSE result and an airspace, finding the probability in the z-direction is straightforward, however the xy-probability is not because the covariance must have a zero off diagonal covariance matrix. A zero off diagonal covariance can be obtained because the covariance needs to be rotated back to the body frame covariance. Since the covariance is rotated, the airspace's 2D polygon must be as well. Once rotated the airspace is discretized into an operator specified amount of grids. Then, the probability of being in a grid that touches or is within the airspace is calculated. Adding each grid's probability yields a conservative xy-probability. Finally, multiplying the xy-probability and z-probability together yields the airspace violation probability.

4.2.1 Rotating the FSE result and airspace

Rotating the 2D covariance matrix appropriately back to the body frame yields a zero off diagonal covariance matrix. The rotation angle back to the body frame is θ, the angle between the ellipse's semi major axis and the +x-body axis. Using a DCM with rotation angle θ transforms the xy-Cartesian covariance back to the body covariance, resulting in a new 2D distribution, with the same

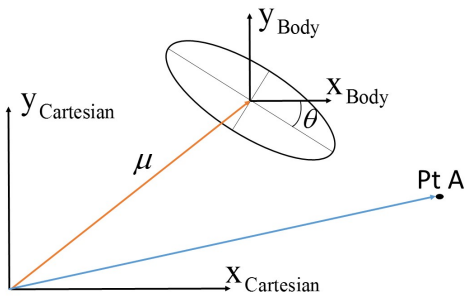

Figure 4.1: Polygon vertex rotation

This shows the relationship between the 2D Gaussian distribution and a polygon vertex (Pt A). Furthermore since the Gaussian distribution needs to be rotated to the body frame, Pt A also needs to be rotated to maintain the same relative distance.

FSE result's 2D distribution's mean and the body covariance [29]. Since the distribution is rotated the airspace must also be rotated via the 2D polygons vertices to maintain the same relative distance between the covariance and the airspace. Figure 4.1 illustrates how a vertex (Pt A) relates to the covariance in both the Cartesian and body frame xy-plane. Let $\begin{bmatrix} x_{vertex} & y_{vertex} \end{bmatrix}^{\mathrm{T}}$ be a polygon's vertex.

A vertex is transformed by

$$\begin{bmatrix} \tilde{x}_{vertex} \\ \tilde{y}_{vertex} \end{bmatrix} = DCM_{Body/XY} \left(\begin{bmatrix} x_{vertex} \\ y_{vertex} \end{bmatrix} - \mu \right) + \mu \tag{4.6}$$

4.2.2 Discretizing the Airspace

After rotating both the 2D Gaussian distribution and the polygon, the polygon is discretized. Discretizing the polygon is necessary to obtain a simplified calculation of the probability of violating the airspace in the xy-plane. The first step is to determine a rectangle which surrounds the polygon. Next this rectangle is discretized based on an operator specified number of blocks in the x-direction and y-direction to create a series of grids within the containing rectangle. The algorithm then determines which grids either touch or are within the polygon. Once all the grids that touch or are within

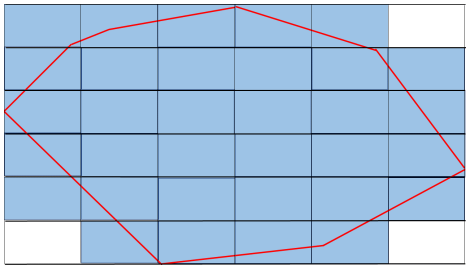

Figure 4.3: Example of a polygon grid domain

The red convex polygon represents a 2D airspace. This polygon is then discretized and the blue grids represents the grids that touch or are within the airspace. These blue grids form the polygon grid domain.

the polygon are found, these grids form the polygon grid domain, where N is the number of grids in the polygon grid domain.

4.2.3 Calculating the Airspace Conflict Probability

Using the grid's maximum and minimum x-value and y-value as the limits of integration, the 2D probability of being in the grid is calculated by Eq.4.4. Let each grid probability be P_{xy_k} where k is the grid number. Thus the probability of being in the polygon grid domain is

$$P_{xy,Tot} = \sum_{k=1}^{N} P_{xy_k} \tag{4.7}$$

Since the polygon grid domain's area is equal or larger than the polygon's area, this implies $P_{xy,Tot} \geq P_{polygon}$. Thus this method is conservative. Examples illustrating this method's conservative nature is discussed next.

Next P_z is calculated using Eq.4.5 where the integration's limits are the airspace's minimum and maximum altitude. Since $P_{xy,Tot}$ and P_z are independent, the total probability is their product. Furthermore $P_{xy,Tot}$ is conservative because $P_{xy,Tot}$ is conservative.

$$P_{tot} = P_z P_{xy,Tot} \tag{4.8}$$

4.3 Airspace Conflict Calculator Validation (ACCV)

The airspace conflict calculator is validated using a Monte Carlo simulation developed by Henry Qin, an undergraduate on the research team. 5000 random positions based on the FSE result are generated. Each point is classified as either violating or not violating the airspace. Then the Monte Carlo result is calculated by dividing the number of violations by 5000. Since the Monte Carlo simulation generates random positions, each simulation results in different conflict probabilities. Thus, this Monte Carlo simulation provides an airspace violation approximation. The airspace conflict calculator validation (ACCV) test passes if the airspace conflict calculator's results are either within 2% of the Monte Carlo results or the airspace conflict calculator results are greater than the Monte Carlo results.

The ACCV test has 13 scenarios, where many scenarios have the airspace conflict calculator's results substantially larger than the numeric results. A large discrepancy occurs because the FSE result's covariance is small compared to the airspace and slightly breaches the airspace. Having these conditions a discretized airspace may have a grid containing the entire distribution leading large probability compared to the Monte Carlo result. All 13 scenarios validate the independence assumption between the 2D and 1D Gaussian and demonstrate that the airspace conflict calculator yields conservative results.

4.3.1 Airspace Conflict Calculator Validation Scenario 1

Scenario 1 has an FSE result approximately halfway breaching the airspace. Thus the Monte Carlo simulation is expected to predict a 50% airspace violation probability. Figure 4.5 illustrates both the scenario and the Monte Carlo simulation result. As expected the Monte Carlo calculates a 48.7% probability, while the airspace calculator predicts an expected 100% probability because a discretized polygon grid contains the 95% confidence region.

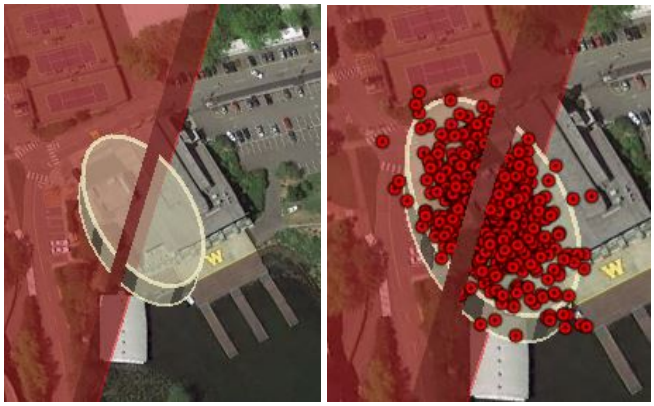

Figure 4.5: ACCV scenario 1, description (left) and Monte Carlo simulation (right)

4.3.2 Airspace Conflict Calculator Validation Scenario 2

Scenario 2 is similar to scenario 1 except the FSE result is larger. Figure 4.6 illustrates both the scenario and the Monte Carlo simulation result. Again, the Monte Carlo simulation yields an expected 48.6% probability, while the airspace conflict calculator predicts a 99.9% probability because a discretized polygon grid contains the FSE result.

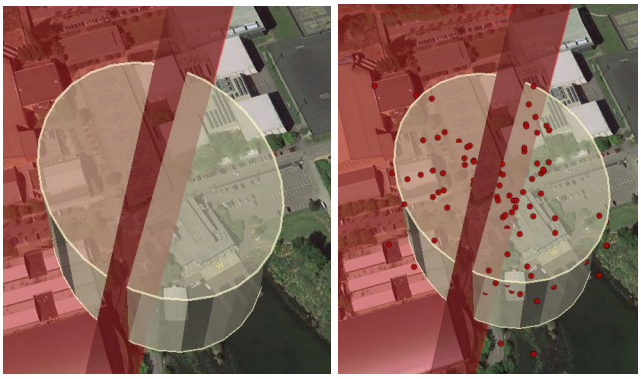

Figure 4.6: ACCV scenario 2, description (left) and Monte Carlo simulation (right)

4.3.3 Airspace Conflict Calculator Validation Scenario 3

Scenario 3 has an FSE result outside the airspace and the discretized polygon. Thus, both the airspace conflict calculator and Monte Carlo simulation should calculate a 0% airspace violation

probability. Figure 4.7 illustrates the scenario and the Monte Carlo simulation result. As expected both the Monte Carlo simulation and the airspace conflict calculator predict a 0% probability.

Figure 4.7: ACCV scenario 3, description (left) and Monte Carlo simulation (right)

4.3.4 Airspace Conflict Calculator Validation Scenario 4

This scenario tests the independence assumption between the 2D and altitude probabilities. A triangular shaped airspace and an FSE result taller than the airspace and a covariance both inside and outside the triangle is used. Since the FSE result is not entirely inside the rectangle containing the airspace, a 100% airspace violation probability is not expected. Also the airspace and FSE result have similar size, the Monte Carlo simulation and the airspace conflict calculator results should be similar. Figure 4.8 illustrates the scenario and Figure 4.9 illustrates the Monte Carlo simulation results. As expected the Monte Carlo simulation yields a 49.1% probability and the airspace conflict calculator has a larger probability of 56.4%.

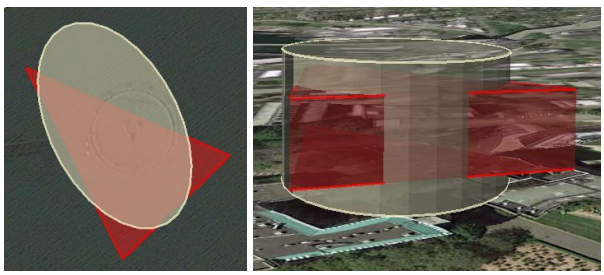

Figure 4.8: ACCV scenario 4, description top view (left) and side view (right)

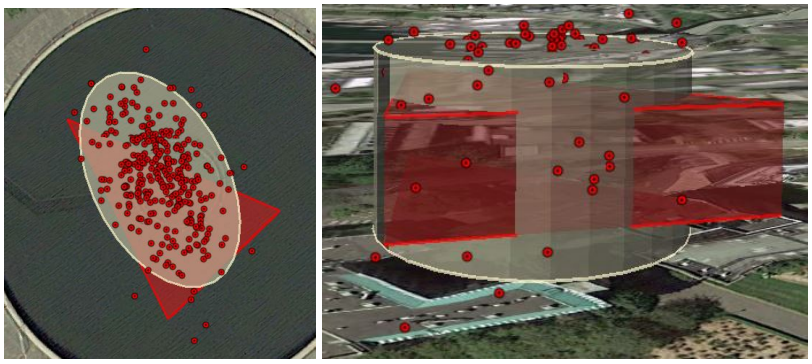

Figure 4.9: ACCV scenario 4, Monte Carlo simulation results top view (left) and side view (right)

4.3.5 Airspace Conflict Calculator Validation Scenario 5-8

The next four scenarios (5-8) involve a restricted airspace and four different FSE results with various sizes and locations as shown in Figure 4.10. For scenarios 5-7 there is an airspace violating FSE result that grows larger after each successive scenario, which represents an aircraft flying towards the restricted airspace. Initially, the FSE result is large as in scenario 7. Despite the warnings, the operator continues flying at the same heading, thus the FSE results become smaller as depicted by scenario 5 and 6. Scenario 8 demonstrates the airspace conflict calculator will not predict a 100% violation because the FSE result has portions outside the rectangle containing the polygon.

For scenario 5 the Monte Carlo simulation (Figure 4.11) yields a 70.1% probability, while the airspace conflict calculator predicts a 100% probability. For scenario 6 the Monte Carlo simulation (Figure 4.12) yields a 56.5% probability, while the airspace conflict calculator predicts a 95.3% probability. For scenario 7 the Monte Carlo simulation (Figure 4.13) yields a 22.2% probability, while the airspace conflict calculator predicts a 62.4% probability. For scenario 8 the Monte Carlo simulation (Figure 4.14) yields a 1.3% probability, while the airspace conflict calculator predicts a 5.2% probability.

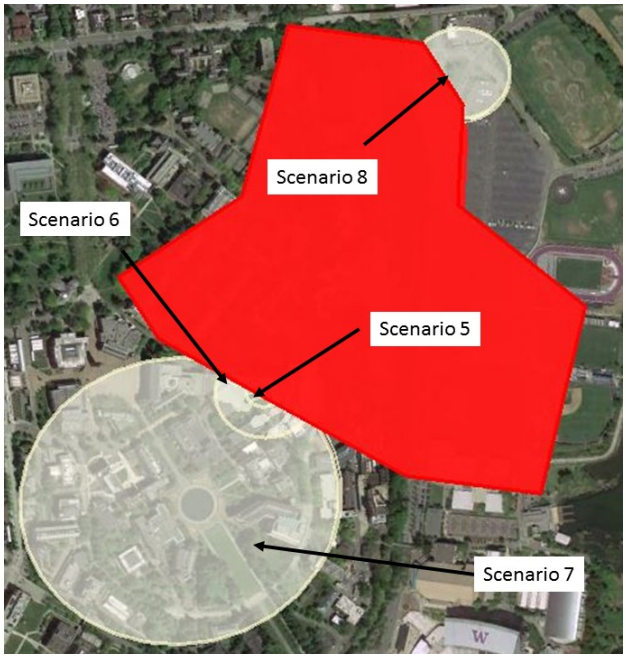

Figure 4.10: ACCV scenario 5-8, overview

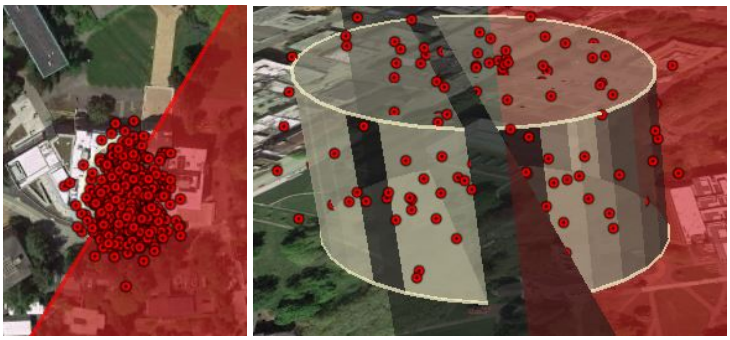

Figure 4.11: ACCV scenario 5, Monte Carlo simulation results top view (left) and side view (right)

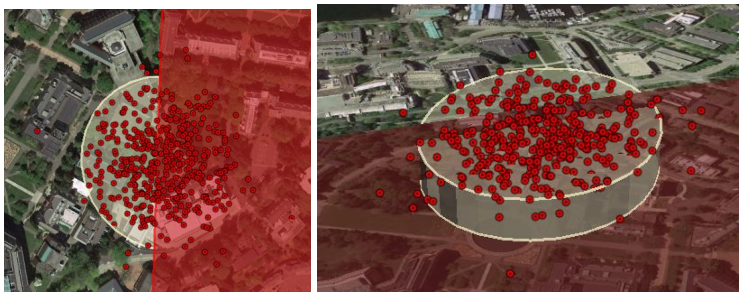

Figure 4.12: ACCV scenario 6, Monte Carlo simulation results top view (left) and side view (right)

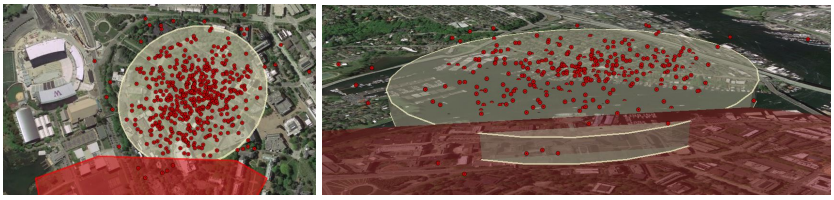

Figure 4.13: ACCV scenario 7, Monte Carlo simulation results top view (left) and side view (right)

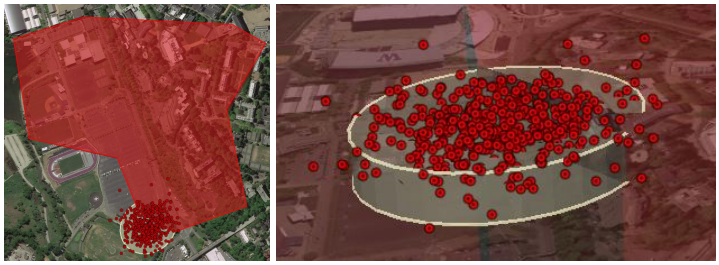

Figure 4.14: ACCV scenario 8, Monte Carlo simulation results top view (left) and side view (right)

4.3.6 Airspace Conflict Calculator Validation Scenario 9-13

Scenarios 9-13 further tests the independence assumption. Figure 4.15 illustrates both the side and top view for scenarios 9-13. Each scenario has an FSE result with varying height resulting in a different intersection percentage however each scenario maintains the same 2D distribution. Therefore the airspace conflict probability will be similar to the percentage the FSE result is in the

airspace. Thus scenario 9 and 11 is expected to have a probability close to 100% while scenario 10 and 12's probability is close to 50% and finally scenario 13 will have a probability close to 0%. Furthermore since the airspace and the FSE result are similar in size, the Monte Carlo simulation and the airspace conflict calculator results should be similar as well.

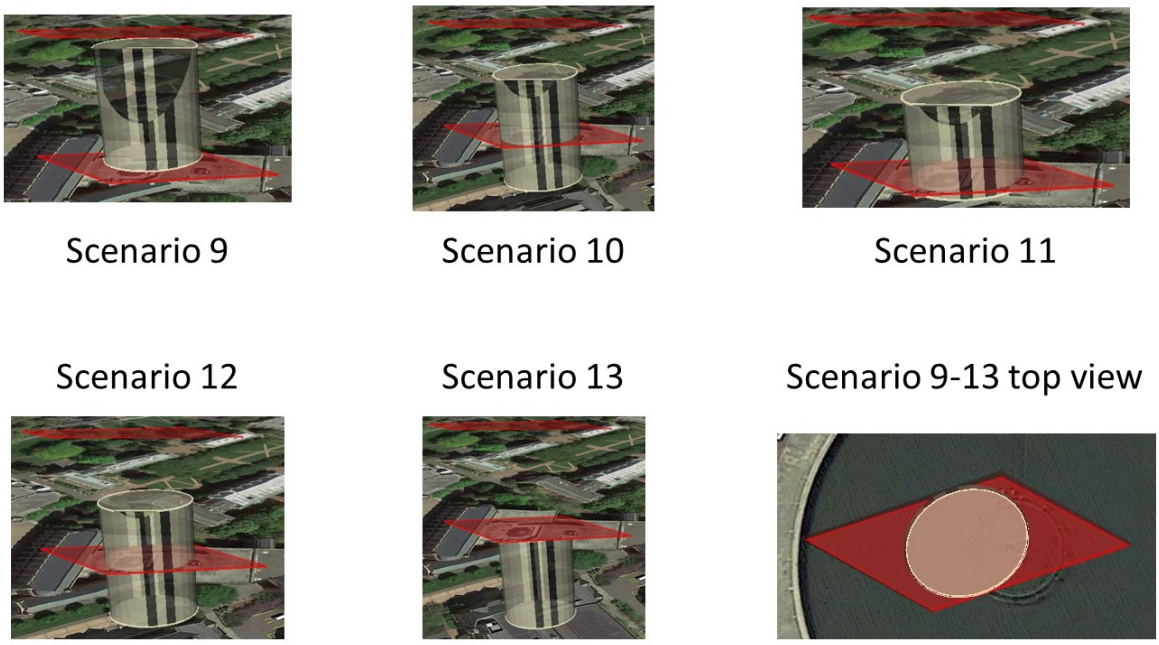

Figure 4.15: ACCV scenario 9-13, overview

For scenario 9 the Monte Carlo simulation (Figure 4.16) yields a 96.8% probability, while the airspace conflict calculator predicts a 96.7% probability. For scenario 10 the Monte Carlo simulation (Figure 4.17) yields a 59% probability, while the airspace conflict calculator predicts a 58.6% probability. For scenario 11 the Monte Carlo simulation (Figure 4.18) yields a 93.7% probability, while the airspace conflict calculator predicts a 93.7% probability. For scenario 12 the Monte Carlo simulation (Figure 4.19) yields a 49.1% probability, while the airspace conflict calculator predicts a 49.4% probability. For scenario 13 the Monte Carlo simulation (Figure 4.20) yields a 2% probability of breaching the aircraft, while the airspace conflict calculator predicts a 2.2% probability.

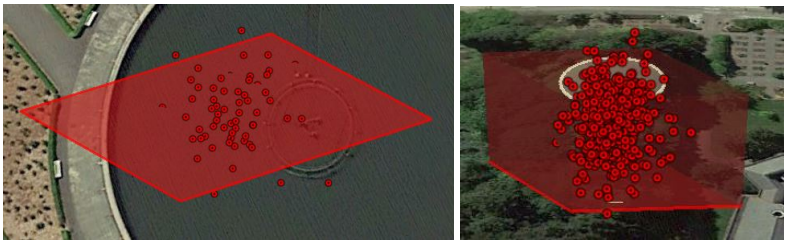

Figure 4.16: ACCV scenario 9, Monte Carlo simulation results top view (left) and side view (right)

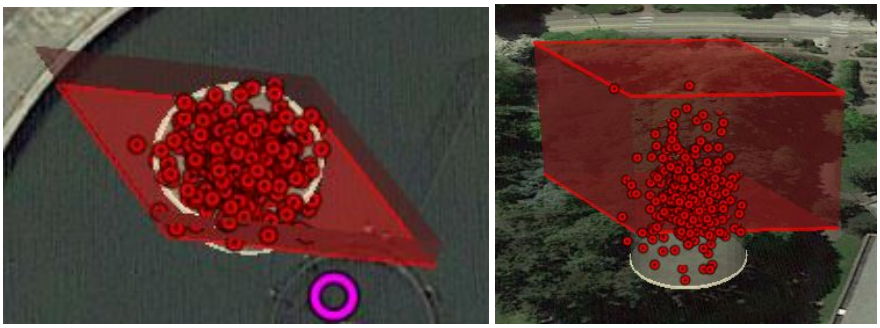

Figure 4.17: ACCV scenario 10, Monte Carlo simulation results top view (left) and side view (right)

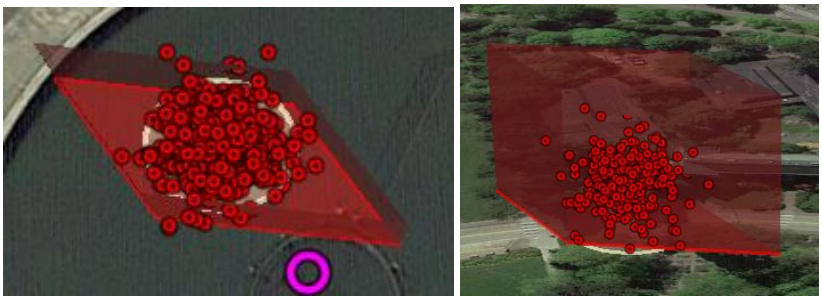

Figure 4.18: ACCV scenario 11, Monte Carlo simulation results top view (left) and side view (right)

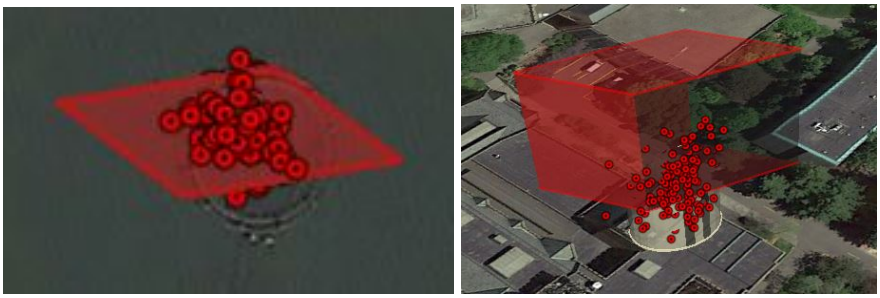

Figure 4.19: ACCV scenario 12, Monte Carlo simulation results top view (left) and side view (right)

Figure 4.20: ACCV scenario 13, Monte Carlo simulation results top view (left) and side view (right)

Chapter 5

SIMULATION

5.1 Introduction

Since ICOMC2 is C# based, the CAPlugin's algorithm is also implemented in C#. Although C# does not contain a large math library, the research team developed many math and CAPlugin related classes to support the forward state estimator and conflict calculator. I helped develop the code for the FSE free flight, FSE orbit and the airspace conflict calculator. The FSE free flight is carefully designed to prevent code redundancy with the FSE orbit. After a class is written, a unit test is created to verify the class functions properly. Once all the unit test pass, different scenarios are engineered to verify the CAPlugin provides reasonable solutions. This chapter discusses the unit test for each class and two complex simulations.

5.2 Unit Tests

Unit tests help debug an individual class' constructor, properties, and public methods. By testing each class individually the programmer can quickly fix the broken code. Complex unit tests were developed to verify the class functions properly and handles the edge cases. Each class is extensively tested because other classes may call on the tested class' public methods. For example, the FSE orbit calls many FSE free flight methods, thus if the FSE free flight provides incorrect results, the FSE orbit will as well. The FSE related classes is tested by comparing the FSE results at different prediction times, while a Monte Carlo simulation verifies the airspace conflict calculator results.

5.2.1 FSE Free Flight Unit Test

Both the FSE free flight and the FSE orbit use the future position, altitude variance and xy-covariance methods, thus these methods require rigorous testing. The FSE free flight is tested by comparing the FSE result at two different prediction times. For example, let an aircraft have a +x-velocity, then the unit test pass if the x-position at 10 seconds is larger than the x-position at 5 seconds.

Since there are many variations between s, σ_s, σ_θ, σ_x and σ_y, calculating the xy-covaraiance is the most complex method and requires extensive testing. Nine different scenarios, including edge cases, were created to capture all the different variations in only the first quadrant. Assuming all the tests in the first quadrant pass, then the other three quadrants will pass as well. The unit test scenarios are similar to the scenarios discussed in the FSE free flight chapter's *Matlab simulation* section.

5.2.2 FSE Orbit Unit Test

Calculating an aircraft's mean position is the most important FSE orbit unit test because there are various scenarios depending on the aircraft initial position. Sixteen different scenarios were written for this unit test, each scenario is similar to the scenarios discussed in the FSE orbit chapter's *Matlab simulation* section.

Another important FSE orbit unit test involves how the FSE orbit limits the covariance method. To test if the covariance is limited, the following logic is used. Let $t_1 > t_{error}$ and $t_2 > t_{error}$, then the covariance's eigenvalues at t_1 must equal the covariance's eigenvalues at t_2 for the unit test to pass. Next let $t_1 < t_2 < t_{error}$, then the covariance's eigenvalues at t_1 must be less than the covariance's eigenvalues at t_2 for the unit test to pass.

5.2.3 Airspace Conflict Calculator Unit Test

The airspace conflict calculator unit test is described in the airspace conflict calculator chapter's *Validating the Airspace Conflict Calculator* section.

5.3 Visualization

Although the CAPlugin will be integrated into ICOMC2, the research team did not receive ICOMC2. Thus, to test the plugin a separate simulator was created. This simulator generates Google Earth (.kml) files, which visualizes the CAPlugin results in Google Earth.

An FSE result is represented as a beige elliptical cylinder which contains the FSE result's 95% confidence level. Furthermore each elliptical cylinder represents a different prediction time, thus an FSE result grows in size as the prediction time increases. When an aircraft is engaged in free flight

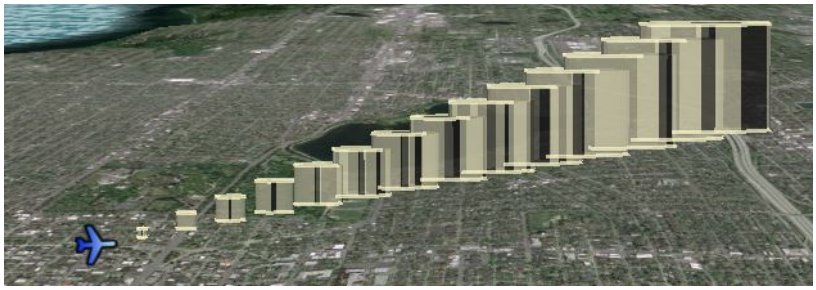

Figure 5.1: FSE free flight example side view

In this FSE free flight example an aircraft is currently climbing as shown by an increase in altitude.

Figure 5.3: FSE free flight example top view

This shows the FSE free flight assumption that the aircraft maintains the current speed and θ angle as the prediction time increases.

mode, only the current aircraft's position and the FSE results will be illustrated. This is illustrated in Figures 5.3 and 5.1. When an aircraft is in orbit mode, the orbit, the current aircraft's position and the FSE results are drawn as illustrated in Figures 5.7 and 5.5. Finally an airspace is illustrated as an extruded 2D convex polygon with vertical side walls (Figure 5.9).

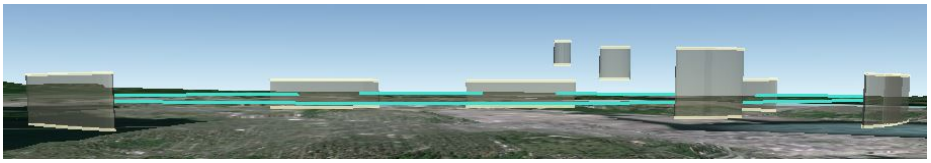

Figure 5.5: FSE orbit example side view

In this FSE orbit example, the aircraft begins above the orbit altitude, thus the aircraft will descend until the orbit altitude is reached. Once the aircraft is at the orbit altitude, the aircraft maintains the orbit altitude.

Figure 5.7: FSE orbit example top view

Since the aircraft is initially on the orbit in the xy-plane but above the orbit altitude, the aircraft will follow the orbit in the xy-plane while descending until the orbit altitude is reached. Once reached the aircraft is assumed to maintain orbit altitude while following the orbit.

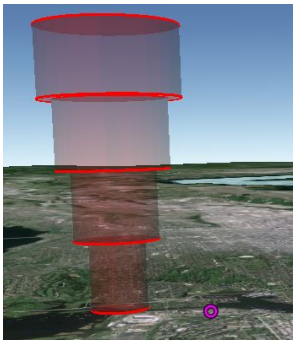

Figure 5.9: Airspace example

In this example this one airspace which contains four separate airspaces stacked on each other.

5.4 Simulation 1

Simulation 1 is a 90 second simulation illustrating the interaction between the FSE and the conflict calculator within the CAPlugin architecture. This scenario contains a restricted airspace and three different aircraft, a UAS (UAS06), another UAS (UAS04), and a jet (Jet01). Initially, both Jet01 and UAS04 are in free flight and below UAS06's altitude. UAS06 is currently following a parallel track flight path. South of UAS06's flight path is the restricted airspace. Figures 5.13 and 5.11 illustrates the initial FSE predictions which includes a 30 second prediction time FSE results. At this time, the conflict calculator predicts no potential conflicts.

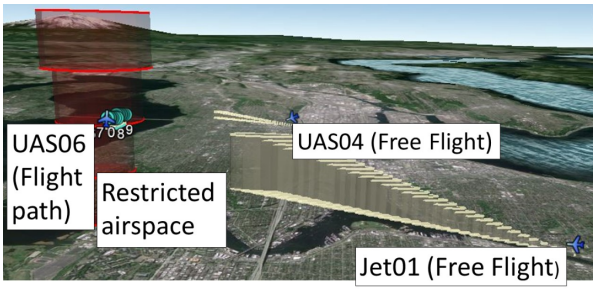

Figure 5.11: Simulation 1, initial FSE results side view

This is a side view of simulation 1's FSE results at 0 seconds. In this scenario, there are 2 aircraft in free flight, 1 aircraft in a flight path mode and an airspace. Forward state estimates are predicted 30 seconds into the future.

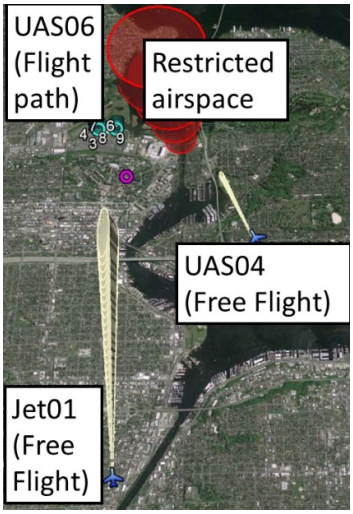

Figure 5.13: Simulation 1, initial FSE results top view

This is a top view of simulation 1's FSE results at 0 seconds. Furthermore this figure better depicts the FSE results for all 3 aircraft.

5.4.1 Time = 10 seconds

In the first 3 seconds there are no potential conflict warnings. However from 4 to 10 seconds, the system predicts Jet01 will be in conflict with UAS06 in the next 30 seconds with 100% probability. Figure 5.15 illustrates the FSE result at 10 seconds. Although the two FSE results do not intersect, the conflict calculator predicts 100% probability due to the conservative methods applied.

5.4.2 Time = 24 seconds

For the next 14 seconds the system warns the operator of the pending conflict with 100% probability and the time of conflict continues to decrease. At 24 seconds the system predicts the time of conflict is 12 seconds. The UAS06 changes to an expanding square flight pattern south of Jet01's predicted position. Figure 5.17 illustrates the FSE results for each aircraft at 25 seconds. UAS06 stays in the expanding square flight path for the next 20 seconds.

Figure 5.15: Simulation 1, FSE results at 10 secs side view (left) and top view (right)

At 10 seconds there is a pending conflict between Jet01 and UAS06. The top view shows the close proximity between the Jet01's FSE result and UAS06's FSE result.

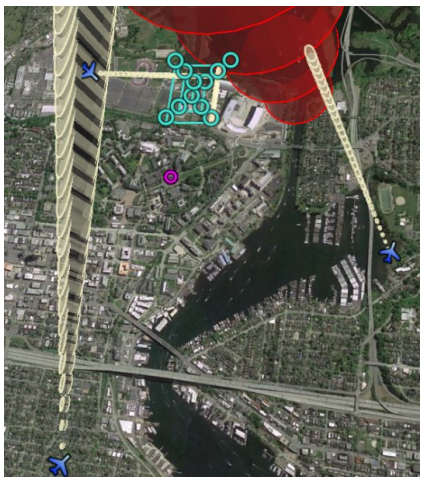

Figure 5.17: Simulation 1, FSE results at 25 secs

At 25 seconds since the conflict between UAS06 and Jet01 is highly probable, UAS06's operator changes the flight path from the parallel track to the expanding square.

5.4.3 Time = 45 seconds

At 45 seconds the UAS04 is still in free flight and Jet01 clears the parallel track as illustrated in Figure 5.19. Since the parallel track is cleared, UAS06 returns to the parallel track and the operator changes the perspective entity to UAS04 and switches UAS04's flight mode from free flight to flight path mode following the expanding square flight route.

5.4.4 Time = 81 seconds

From 51 seconds to 80 seconds the system predicts a conflict between the UAS04 and UAS06 with 100% probability. These warnings are expected because the expanding square and parallel track flight route are within the horizontal separation distance, which further illustrates the conservative methods used by the aircraft conflict calculator. However at 81 seconds the system warns that UAS04 is currently violating the restricted airspace as illustrated in Figure 5.21. These warnings remain for the remainder of the simulation.

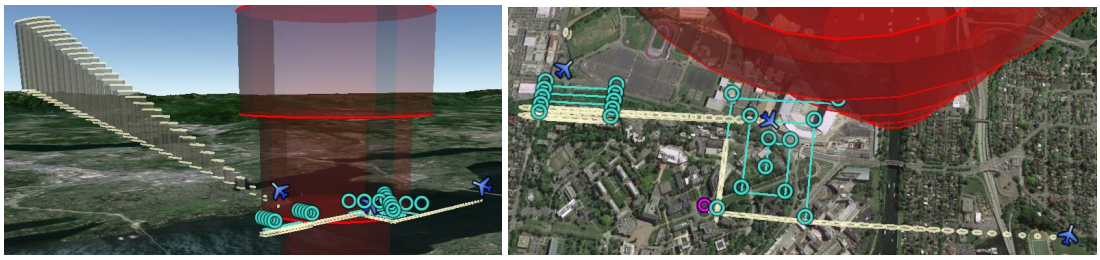

Figure 5.19: Simulation 1, FSE results at 45 secs side view (left) and top view (right)

At 45 seconds Jet01 clears the parallel track, thus UAS06's operator switches UAS06's flight path back to the original parallel track flight path. Next the UAS06's operator relinquishes command of UAS06 after sending UAS06 back to the parallel track. Next the operator changes perspective entity and controls UAS04. This operator then switches UAS04's flight mode from free flight to following the expanding square flight path.

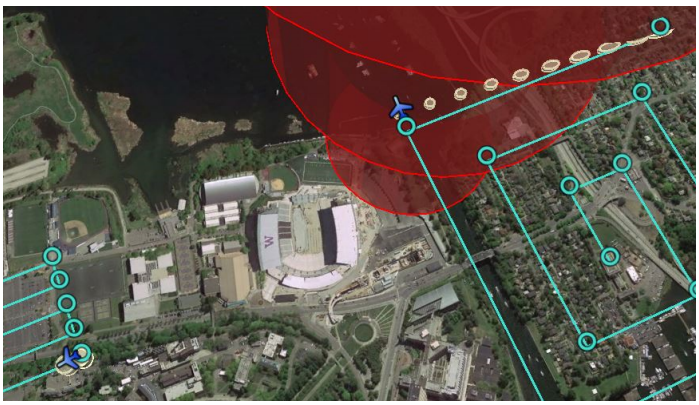

Figure 5.21: Simulation 1, FSE results at 81 secs

At this time the airspace conflict calculator determines UAS04 is currently violating the airspace.

5.5 Simulation 2

Simulation 2 illustrates the collision awareness algorithm capabilities to handle a free flight general aviation aircraft (Prop13)). In this simulation Prop13 will be in conflict with UAS05, an orbiting UAS, and a restricted airspace. This simulation lasts 60 seconds with a 20 second prediction time. Figure 5.23 illustrates the initial FSE results. At 0 seconds the conflict calculator predicts UAS05 will breach Prop13's airspace in 14 seconds with 100% probability. As expected the system warns Prop13 about the pending conflict with UAS05 for the first 14 seconds because the FSE results are in close proximity to each other.

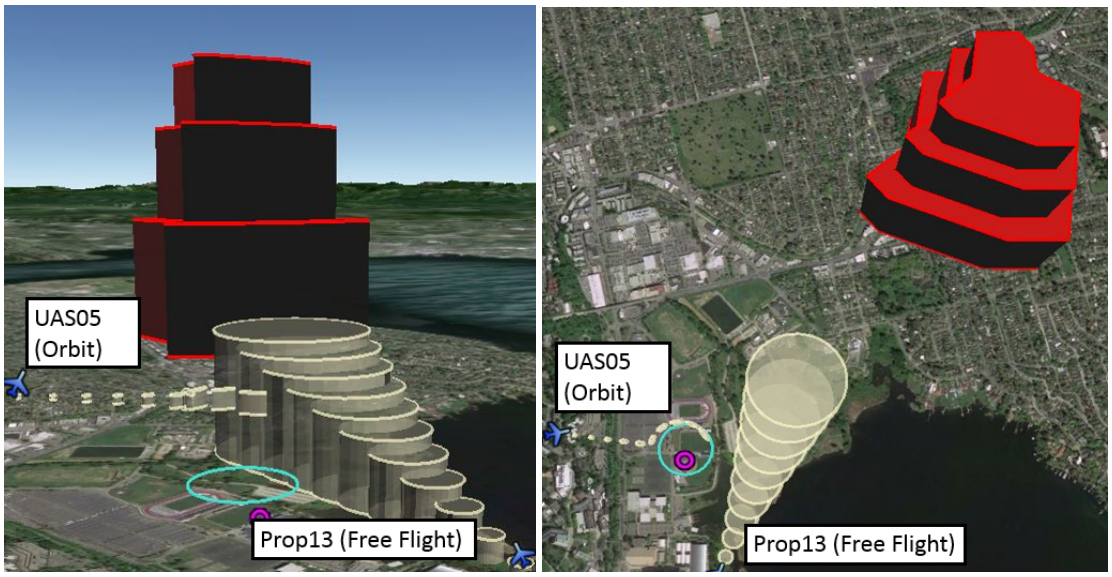

Figure 5.23: Simulation 2, initial FSE results

This is a side view of simulation 2's FSE results at 0 seconds. In this scenario, there is 1 aircraft in free flight, 1 aircraft in orbit mode and an airspace. Furthermore the initial FSE results between UAS05 and Prop13 are within close proximity and the aircraft conflict calculator predicts conflicts between these two aircrafts for the next 14 seconds.

5.5.1 Time = 14 seconds

As the simulation progresses the conflict calculator continues to predict that Prop13's airspace will be breached with 100% probability. From 8 to 12 seconds the conflict calculator predicts that UAS05

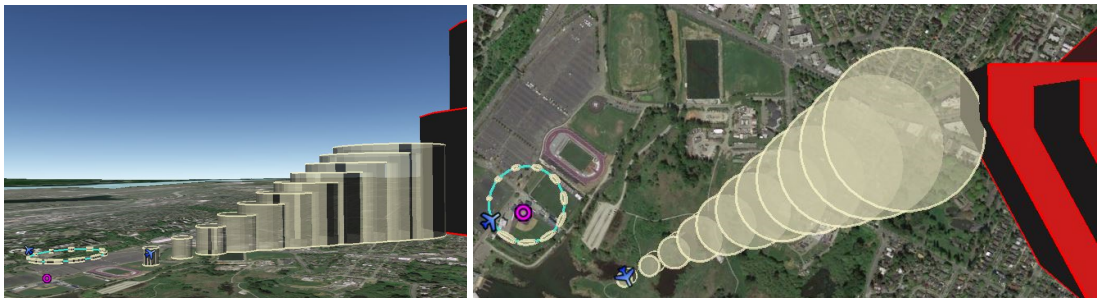

Figure 5.25: Simulation 2, FSE results at 14 seconds side view (left) and top view (right)

At 14 seconds the warnings between the potential conflicts between Prop13 and UAS05 ceases. The system then warns the operator of a pending airspace violation.

has breached Prop13's airspace with 100% probability. Once Prop13 pasts UAS05's orbit, the system predicts a potential conflict with the airspace as illustrated in Figure 5.25. The conflict calculator predicts a 6.76% probability of violating the airspace in 10 seconds at 14 seconds. As the simulation progresses the conflict calculator continues to predict a violation with increasing probability. For instance at 20 seconds the probability of violation is 61.15% and at 32 seconds the probability of violation is 89.32%.

5.5.2 Time = 32 seconds

At 32 seconds there is an interesting result because the probability of violation is 89.32% in 8 seconds however Figure 5.27 shows the FSE result does not appear to be contained by 89.32% of the airspace. This discrepancy is caused by the conservative methods implemented in the airspace conflict calculator. Having an over conservative method is important because the pilot would not continue to fly the current heading knowing a high probability of an airspace violation exists.

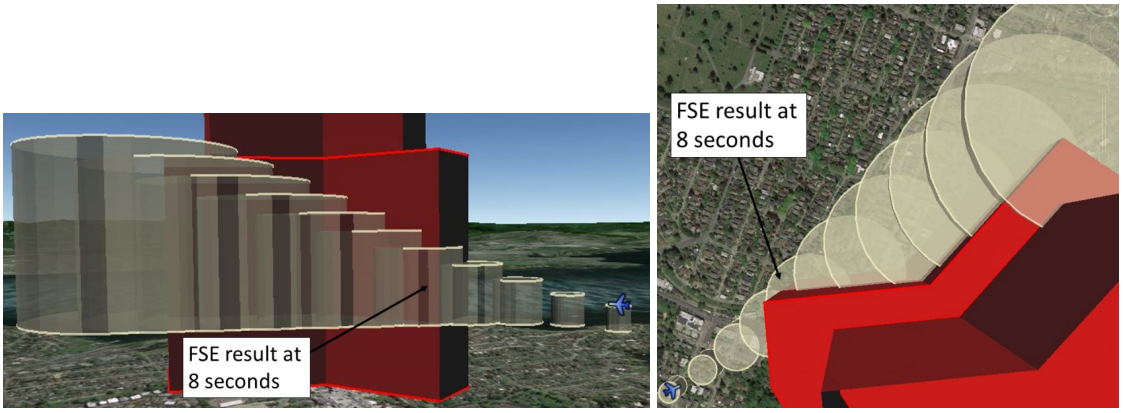

Figure 5.27: Simulation 2, FSE results at 32 seconds side view (left) and top view (right)

At 32 seconds the airspace conflict calculator predicts an 89.32% probability of violating the airspace occurring in the next 8 seconds.

5.5.3 Time = 40 seconds

Figure 5.29 shows the FSE results of the aircraft and the airspace at 40 seconds. At this time the conflict calculator predicts that Prop13 is violating the airspace with 100% probability.

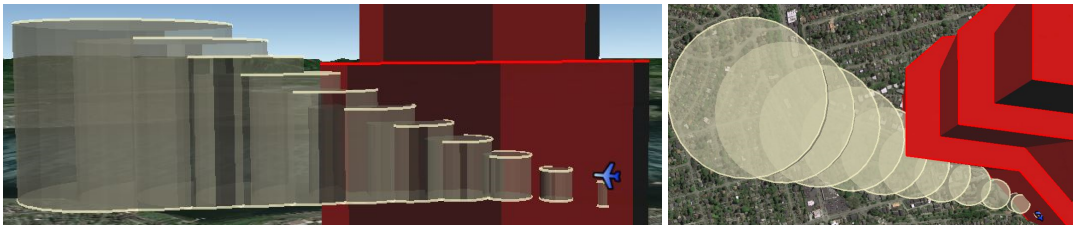

Figure 5.29: Simulation 2, FSE results at 40 seconds side view (left) and top view (right)

At this time the aircraft is violating the airspace as predicted by the airspace conflict calculator.

The conflict calculator continues to predict with a probability close to a 100% that Prop13 is violating the airspace. At 52 seconds Prop13 has cleared the airspace and the conflict calculator has no conflict warnings for the remainder of the simulation.

Chapter 6

CONCLUSION

The algorithms for the FSE free flight, FSE orbit, and airspace conflict calculator are simple, overly conservative and easy to implement. Since the FSE models an aircraft as a 3D point mass, any UAS or manned aircraft's position can be predicted. This assumption yields a closed form solution, thus prediction calculations are done quickly and require minimum memory. Also the FSE predictions are current and are updated at least every second because aircraft information is provided at least every second according to STANAG 4586 [30], a NATO document describing UAS standards complied with by NATO nation's military forces. The airspace conflict calculator overestimates the airspace's 2D area, thus calculating a conservative conflict probability.

6.1 FSE Free Flight

Although the mean is calculated using a 3D point mass model, the covariance calculation makes the FSE free flight conservative. Since a second order error propagation method is used this calculates a conservative results as discussed in the FSE free flight chapter. Another conservative assumption is applying the same speed used in the mean calculation to calculate the covariance and variance.

6.2 FSE Orbit

The FSE orbit accounts for situations when the aircraft is not on the orbit. Although the assumptions that model how an aircraft reaches an orbit do not reflect normal operating procedures, the algorithm proposed sets the ground work necessary to developing such an algorithm. However, modeling an aircraft's motion on the orbit is accurate when the aircraft starts on the orbit. Also the covariance calculations and associated assumptions yield reasonable results.

6.3 Airspace Conflict Calculator

The airspace conflict calculator provides the CAPlugin with a conservative airspace violation probability because the integration area is larger than the 2D polygon's area. Furthermore the airspace conflict calculator does account for the worse case scenario by integrating the entire probability distribution and not limiting the distribution to a specified confidence level. To obtain the actual probability of an airspace violation the conflict calculator would require an infinite number of blocks. Since this is not realistic, the airspace conflict calculator's algorithm will always provide a conservative probability. Furthermore, the Monte Carlo simulation's results show the airspace conflict calculator's results are conservative.

6.4 Future Work

Since the FSE free flight and airspace conflict calculator algorithm already provide conservative results, there are no foreseeable improvements for these two algorithms. However the FSE orbit can be vastly improved. The first improvement is providing realistic FSE results by modeling the UAS operator's standard procedures when the aircraft is not on the orbit. Modeling an aircraft entering the orbit tangentially yields realistic results. Also by modeling an aircraft, not at the orbit altitude, descending and ascending at the max climb and descent angle provides realistic results. The second improvement is supporting elliptical orbits.

Improving the CAPlugin's warning system will greatly increase the operator's situational awareness. Currently, the system only provides information on the highest conflict probability, but does not warn the operator if there are other highly probable conflicts. This is demonstrated in simulation 1, when UAS04 is following the expanding square route, the system only warns about the conflict with UAS06 while there is a clear airspace violation in the future. Thus, if the system can warn the operator of all the potential conflicts, this will vastly improve the operator's situational awareness. This algorithm should consider the conflict probability, time until conflict and collision severity between the aircraft and the other entity.

BIBLIOGRAPHY

[1] C.W. Lum, J. Vagners, and R.T. Rysdyk. Search algorithm for teams of heterogeneous agents with coverage guarantees. *AIAA Journal of Aerospace Computing, Information, and Communication*, 7:1–31, January 2010.

[2] C.W. Lum and J. Vagners. A modular algorithm for exhaustive map searching using occupancy based maps. In *Proceedings of the 2009 Infotech@Aerospace Conference*, Seattle, WA, April 2009.

[3] C.W. Lum, J. Vagners, J.S. Jang, and J. Vian. Partitioned searching and deconfliction: Analysis and flight tests. In *Proceedings of the 2010 American Control Conference*, Baltimore, MD, June 2010.

[4] Federal Aviation Administration. Fact sheet - unmanned aircraft systems (UAS), 2014.

[5] U.S. Department of Transportation Federal Aviation Administration. Integration of civil unmanned aircraft systems (UAS) in the national airspace system (NAS) roadmap, 2013.

[6] C. W. Lum, M. Vavrina, J. Vagners, and J. Vian. Formation flight of swarms of autonomous vehicles in obstructed environments using vector field navigation. In *Proceedings of the 2012 International Conference on Unmanned Aircraft Systems*, June 2012.

[7] C.W. Lum and R.T. Rysdyk. Time constrained randomized path planning using spatial networks. In *Proceedings of the 2008 American Control Conference*, Seattle, WA, June 2008.

[8] C.W. Lum, M.L. Rowland, and R.T. Rysdyk. Human-in-the-loop distributed simulation and validation of strategic autonomous algorithms. In *Proceedings of the 2008 Aerodynamic Measurement Technology and Ground Testing Conference*, Seattle, WA, June 2008.

[9] M.L. Cummings, A.R. Kirschbaum, A. Sulmistras, and J.T. Platts. Stanag 4586 human supervisory control implications. *Air and Weapon Systems Department, Dstl Farnborough & the Office of Naval Research*, 2006.

[10] T. McGeer. Aerosonde hazard estimation. Aerovel Corporation, 1994.

[11] J.N. Anno. Estimate of Human Control Over Mid-Air Collisions. *Journal of Aircraft*, 19(1):86–88, 1982.

[12] R.A. Clothier, R.A. Walker, N. Fulton, and D.A. Campbell. A casualty risk analysis for unmanned aerial system (UAS) operations over inhabited areas. In *Proceedings of Twelfth Australian International Aerospace Congress, 2nd Australasian Unmanned Air Vehicles Conference*, Melbourne, March 2007.

[13] C.W. Lum and B. Waggoner. A risk based paradigm and model for unamnned aerial vehicles in the national airspace. In *Proceedings of the 2011 Infotech@Aerospace Conference*, St. Louis, MO, March 2011.

[14] C.W. Lum, K. Gauksheim, T. Kosel, and T. McGeer. Assessing and estimating risk of operating unmanned aerial systems in populated areas. In *Proceedings of the 11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, Virginia Beach, VA, September 2011.

[15] T.S. Bruggemann and L. Mejias. Airborne collision scenario flight tests : impact of angle measurement errors on reactive vision-based avoidance control. In Brian Falzon, editor, *15th Australian International Aerospace Congress (AIAC15)*, Melbourne, VIC, February 2013.

[16] J. Lai, J.J. Ford, L. Mejias, and P. O'Shea. Characterization of sky-region morphological-temporal airborne collision detection. *Journal of Field Robotics*, 30(2):171–193, March 2013.

[17] Federal Aviation Administration. Introduction to TCAS II. Technical report, FAA, February 2011. version 7.1.

[18] Talked with air traffic controllers and toured Seattle Center with Matt McCully, August 2013.

[19] Insitu Inc. ICOMC2, 2013.

[20] Joint Center For Aerospace Technology Innovation. About, 2013.

[21] D. Accardo, G. Fasano, L. Forlenza, A. Moccia, and A. Rispoli. Flight test of a radar-based tracking system for UAS sense and avoid. *IEEE Transactions on Aerospace and Electoronic Systems*, 49(2):1139–1160, April 2013.

[22] D.S. Duggan, D.A. Felio, and C.S. Askew. Autonomous collision avoidance system for unmanned aerial vehicles, February 19 2013. US Patent 8,380,425.

[23] C. Luo, S.I. McClean, G. Parr, L. Teacy, and R. De Nardi. UAV position estimation and collision avoidance using the extended Kalman filter. *Vehicular Technology, IEEE Transactions on*, 62(6):2749–2762, July 2013.

[24] C.E. Lin, Y. Lai, and F. Lee. UAV collision avoidance using sector recognition in cooperative mission to helicopters. In *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2014*, pages F1–1 – F1–9, Hernodon, VA, April 2014.

[25] V. Cichella, R. Choe, S.B. Mehdi, E. Xargay, N. Hovakimyan, V. Dobrokhodov, and I. Kaminer. Trajectory generation and collision avoidance for safe operation of cooperating UAVs. In AIAA Sci Tech, editor, *AIAA Guidance, Navigation, and Control Conference*, 2014.

[26] M. Peralta. *Propagation of Errors: How to Mathetamically Predict Measurement Errors*. CreateSpace, second edition, 2013.

[27] Amy Arbeit, Insitu aircraft operator and programmer. Interview, April 2014.

[28] Insitu Inc. FSE orbit discussion, June 2014.

[29] K.K. Ueunten, C.W. Lum, A. Creigh, and K. Tsujita. Conservative algorithms for automated collision awareness for multiple unmanned aerial systems. In *IEEE Aerospace Conference*, 2015.

[30] NATO Standardization Agency. STANAG 4586 (Edition 3) - Standard Interfaces of UAV Control System (UCS) For NATO UAV Interoperability, November 2012.