UAV Collision Avoidance:

A Specific Acceleration Matching Approach

Amir Patel

Thesis presented for the degree of Masters of Science in Engineering
In the Department of Electrical Engineering
University of Cape Town

Supervisor: Dr. Simon Winberg

December 2011

DECLARATION

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own.

Signature of Author . . . . . .

31 August 2011

ABSTRACT

An increased level of autonomy is required for future Unmanned Aerial Vehicle (UAV) missions. One of the technologies required for this to occur is an adequate sense and avoid system. A sense and avoid system ensures that the UAV can detect threat aircraft and take evasive action if required. This thesis investigates a collision avoidance system to satisfy a significant portion of the requirements for sense and avoid.

An extensive literature review was performed and comparisons were made. It was hypothesised that a recently published method of UAV guidance, Specific Acceleration Matching (SAM) Control, could address the shortcomings of the current implementations. Additionally, a novel algorithm, the Linear 3D Velocity Guidance Control Algorithm (3DVGC) was developed to address the particular requirements of UAV collision avoidance. The SAM Controllers, 3DVGC and a conventional Collision Cone algorithm were integrated into the first Collision Avoidance System using Specific Acceleration Matching (CASSAM V.1). This system was tested in simulation and although all threats were evaded, this was done by rapid diving manoeuvres which were deemed dangerous.

Further investigation revealed that the guidance problem was in fact nonlinear. Subsequently, a new Nonlinear 3DVGC was developed. Furthermore, the conventional Collision Cone algorithm was extended to a new algorithm, the Projected Collision Avoidance Algorithm (PCCA). The PCCA, SAM Controllers and Nonlinear 3DVGC were integrated into the second version of the system, CASSAM V.2. Specific test scenarios were simulated as well as Monte Carlo simulation being performed. The simulation results were analysed and were broadly supportive of the viability of this architecture for use in future sense and avoid systems.

ACKNOWLEDGEMENTS

I would like to extend my most sincere gratitude to The All Mighty for giving me the strength and patience I needed. I would also like to thank my parents and sister for their unfailing support.

I acknowledge Dr. Simon Winberg for his guidance and for supervising me during this project. Thank you for your encouraging words and assistance.

I am grateful for Tellumat Defence for giving me the opportunity to undertake this challenge and financially supporting this research. My deepest gratitude goes to Mr. Chris Williams for encouraging me to do my Masters and for always aiding me. I would also like to give thanks to Mr. Simon Norval for his continuous support and guidance. The long hours we spent discussing ideas and brainstorming were invaluable. I would also like to convey thanks to Mr. Dave Jackson for teaching me about “real aeroplanes” and always encouraging me. Additionally, to all my colleagues at Tellumat: a great thank you for always being helpful and empathetic.

I acknowledge Dr. Iain Peddle and Mr. Deon Blaauw for teaching me all I know about UAVs and Control Systems in general. Thanks for making me an honorary Control System Warrior!

Mr. Shuaib Omar, thank you for making the late nights bearable and for always being keen to bounce ideas around. To you I say, “Cheers!”

Finally, my dearest Ilhaam Dalwai, thank you for being the kindest person I know. Your boundless patience and support always gave me hope.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

This thesis documents a research and development project focusing on the development of a Collision Avoidance System (CAS) which can be integrated into a larger Sense and Avoid System (SAS) for an Unmanned Aerial Vehicle (UAV).

1.1 Background

The age of the Unmanned Aerial Vehicle (UAV) has begun. In recent years, the Unmanned Aerial Vehicle industry has undergone extreme growth and become an invaluable force in modern airspace. This has never been more prominent than in the military sector where these aircraft perform complex tasks such as Intelligence, Surveillance and Reconnaissance (ISR), Battle Damage Assessment (BDA) and Fire Correction [1]. In addition, there are also several civilian applications such as Border Patrol and Coastal Management which are well suited to UAVs [2]. An example of such an application is illustrated in Figure 1 by the General Atomics *Predator* UAV performing Border Patrol.

Figure 1: Predator performing surveillance on the southern border of the USA.

However, future (more advanced) missions will require UAVs to operate in close vicinity to other aircraft in the airspace. This presents a safety problem, as currently UAVs do not possess a collision avoidance capability. In contrast,

even when manned aircraft are in flight and under Air Traffic Control (ATC), the last line of defence is still the pilot's vision or his ability to *See & Avoid* other aircraft [3]. This is graphically shown in Figure 2.

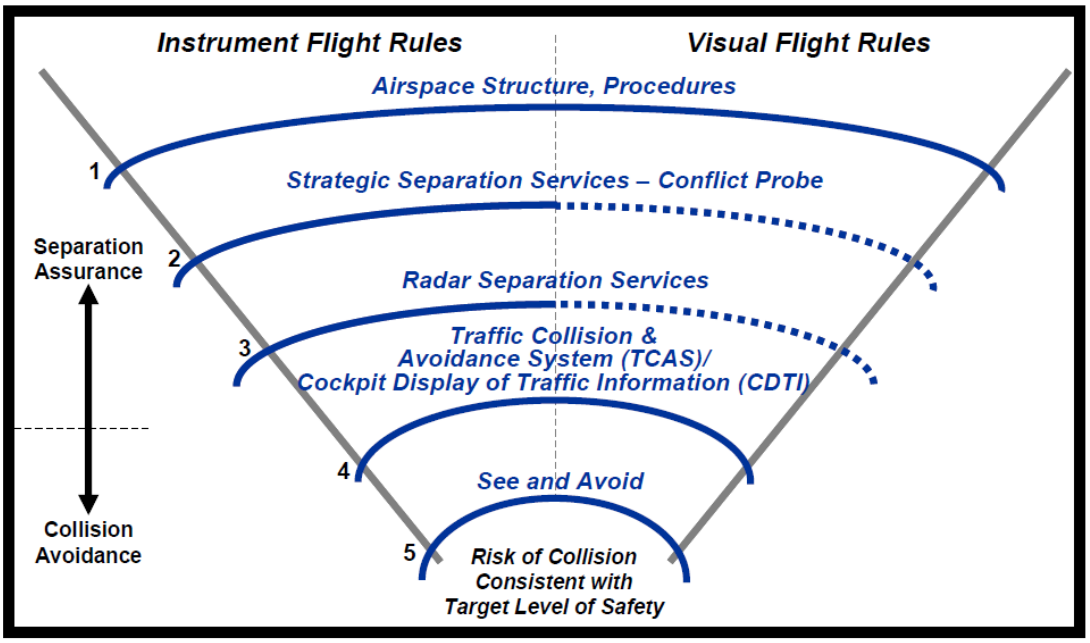

Figure 2: National Airspace layered approach to avoiding collisions. Notice that See and Avoid capability is the lowest level in the hierarchy [3].

This briefly demonstrates why a see & avoid (or rather a *sense*[1] & avoid) capability is on the critical path for the future development of UAVs. In fact, the US Department of Defence considers it to be one of the enabling technologies for the future of Unmanned Systems [1].

1.2 Sense and Avoid Concept

In October 2009, the Federal Aviation Administration (FAA), the national aviation authority of the United States, sponsored a workshop where experts and key role players in the UAV fraternity met to discuss the sense and avoid dilemma [4]. In what follows, a brief overview of this document is given.

The main themes discussed in the workshop were the following:

[1] This has become the standard term to define See and Avoid for UAVs as to emphasize the fact that vision is not necessarily being employed by the system [3].

- Sense and avoid for UAVs was defined.
- Key Concepts were defined.
- Functions and Sub-Functions required of a sense and avoid system (SAS) were defined.
- Roles and Responsibilities were discussed.
- Evaluation Methods and Metrics for sense and avoid were discussed.

According to [4], sense and avoid is defined as the capability of a UAS to remain "well clear" and avoid collisions with other traffic. The primary functions of a SAS are: Self Separation and Collision Avoidance. The applicable definitions which resulted from the workshop are given below[2]:

- ***Self Separation*** **(SS)** is defined as the function where the UAV performs manoeuvres to prevent activation of the collision avoidance function while maintaining air traffic separation distance standards. In certain classes of airspaces, this function is performed by ATC.

- ***Collision Avoidance*** **(CA)** is the function of last minute manoeuvring to prevent penetration of the Collision Volume. Collision avoidance becomes necessary when methods (ATC, Rules-of-the-Air, Self Separation, etc.) have failed.

- **The Collision Volume** (**CV**) is comprised of a cylinder with a radius of 500 feet and vertical a height of 200 feet. Any penetration of this volume can be considered to be a Near Mid-Air Collision (NMAC). The reader should note that even though the aircraft may not directly collide

[2] The reader should note, there are several other definitions presented in the workshop report [4] and is encouraged to browse it for further insight.

with the UAV, if the CV is breached the UAV will most certainly be caught in wake turbulence[3].

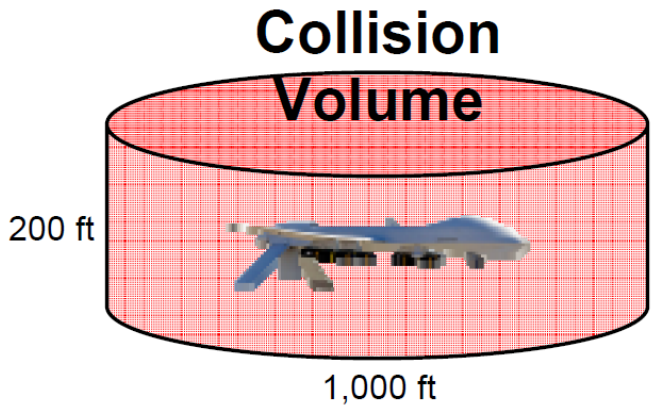

Figure 3: Collision Volume around UAV [4].

- **Cooperative Aircraft** are aircraft which possess an operational transponder for identification purposes. These transponders can be used for cooperative collision avoidance by systems such as TCAS [5].
- **Non-Cooperative Aircraft** are aircraft which do not have a transponder on board or the transponder has malfunctioned or been deliberately disengaged.

The workshop also discussed the sub-functions that a SAS should have the ability to perform. These are the following:

1. **Detect** – Sense the presence of threats (aircraft and other obstacles) in its vicinity.
2. **Track** – Estimate position and velocity of the threat aircraft based on sensor observations.
3. **Evaluate** – Assess the likelihood of collision with the threat based on UAV and threat states.
4. **Prioritise** – Determine which threat poses the greatest risk.
5. **Declare** – Declare that an action is needed by the UAV.

[3] Wake Turbulence is defined as turbulence which is generated by the passage of an aircraft in flight.

6. **Determine Action** – Decide which action or manoeuvre to perform.
7. **Command** – Convey the action to the UAV.
8. **Execute** – The UAV will execute the commanded action or manoeuvre.

These sub-functions are illustrated graphically in a typical encounter scenario below in Figure 4.

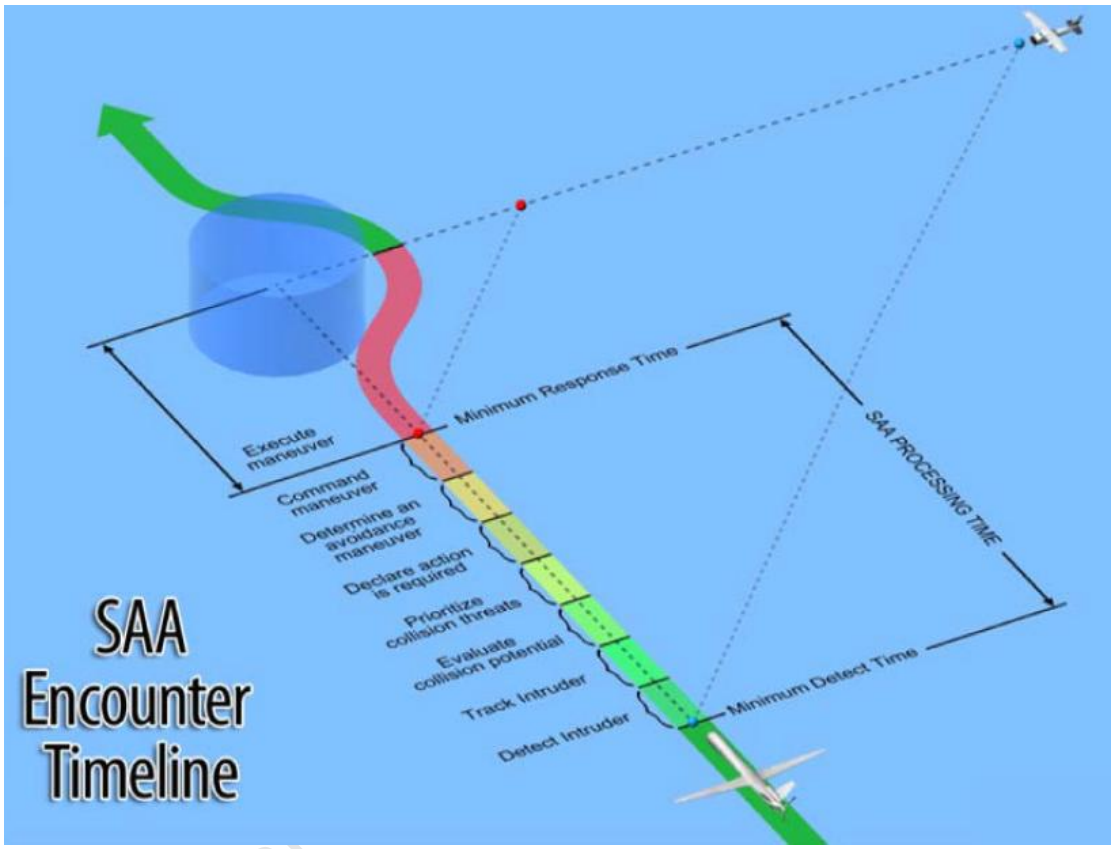

Figure 4: A typical threat encounter is shown. Note the sub-functions are also illustrated [4].

Another important statement from [4] is that the SAS is not envisioned to return the UAV back to the original mission once the threat has been avoided.

There has also been a NATO report released on the matter of sense and avoid for UAVs [6]. It can be considered as a first attempt towards establishing a standard for sense and avoid systems. The report presents a series of requirements, most notable of which is that the SAS should ensure

that the **Probability of Mid-Air Collision per flight hour should not exceed $1X10^{-9}$**. This is the Equivalent Level of Safety (ELOS) to manned aircraft.

However, the workshop [4] states that this level of safety is “inapplicable” for SAS’s for UAVs as the risk of collision would vary for different classes of airspace, threat aircraft, altitudes, etc.

Additionally, according to a study by the National Transport Safety Board [7], the majority of Near Mid-Air Collisions occur under VFR[4] conditions. This information will prove beneficial when designing a SAS for UAVs.

The NATO Report [6] also specifies a minimum Field of Regard (FOR) requirement for the collision avoidance system. The document specifies a minimum of ±110° in azimuth with respect to the longitudinal axis of the UAV and a minimum elevation FOR of ±15°.

The sense and avoid problem for UAVs has now been put into context for the reader. In what follows, the scope and objectives for this particular thesis will be specified.

1.3 Project Scope and Outcomes

Using the combination of both the FAA workshop [4] and the NATO report [6], the thesis scope can be defined. Figure 5, gives an overview of the functionality required for a SAS. However, this thesis will only handle a subset of these functions. Thus, if we consider each of the functions as a “block” of capability, only the blue blocks will be investigated.

[4] VFR or *Visual Flight Rules* are set of regulations which allow the pilot to fly an aircraft in weather conditions clear enough to see the surrounding airspace.

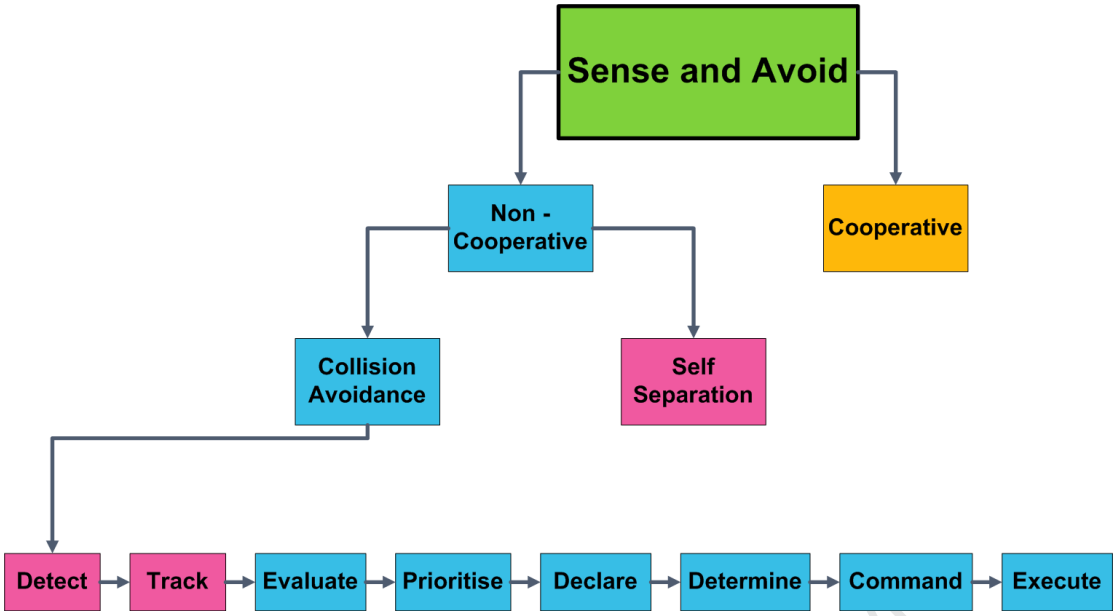

Figure 5: Sense and avoid functional requirements. Blue blocks fall within the scope of this thesis.

In summation, the outcomes of this thesis are:

- Design of an autonomous[5] collision avoidance system (CAS) for fixed-wing UAVs. This system will consist of a suite of algorithms satisfying the aforementioned capabilities.
- A thorough literature survey of various algorithms to benefit from the latest technology.
- Demonstrate a CAS capable of guiding the UAV to safety in a single threat encounter scenario.
- Develop performance metrics to test the effectiveness of the CAS.

The key assumptions are:

- The threat will be assumed to be non-manoeuvrable (constant velocity), non-cooperative and capable of moving in 3D.
- It will be assumed that another set of algorithms exists which will detect and track the threat. These will be transparent to the CAS.

[5] This makes reference to the SAA Workshop [4] discussion on Roles and Responsibilities. Autonomous is selected as opposed to Man-in/on-the Loop.

- The CAS will not be required to return the UAV to its flight path after the threat has been evaded.
- The system will not be tested on hardware but will be tested in simulation. This was done to account for the inherent safety implications of testing on an actual UAV.

A final note is that the algorithms should factor in the limitations of the aircraft with regards to its manoeuvrability.

1.4 Thesis Outline

This thesis is structured into three main themed sections. The first three chapters can be considered as the *Preamble* section where the reader is given insight and perspective into the sense and avoid problem and also how this thesis plans on addressing it. This leads into the *Modelling and Design* section, which consist of four chapters. The final section is the *Interpretation* section which consists of the final two chapters of the thesis. An overview is presented in Figure 6.

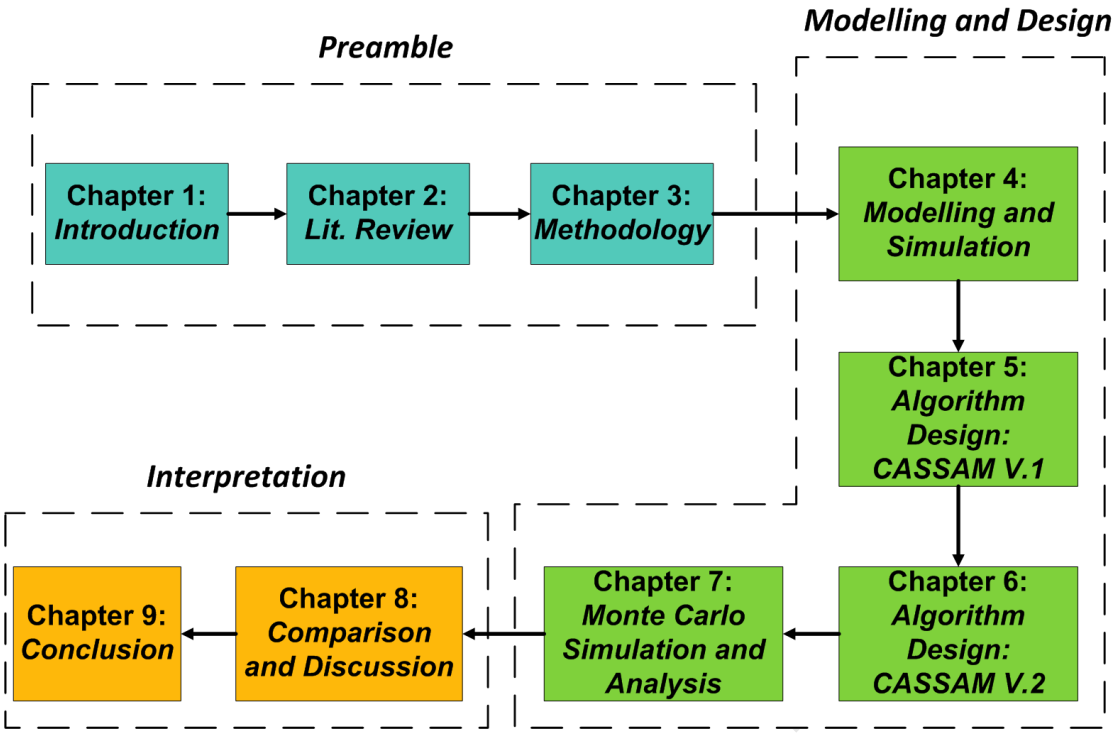

Figure 6: Thesis progression and outline.

In **Chapter 2**, a survey into the literature pertaining to the field of study is given. This will serve as a solid foundation on which the algorithms in the subsequent chapters will be designed.

In **Chapter 3**, the design methodology is described. In particular, the design and research process followed is explained in detail.

In **Chapter 4**, the simulation environment is designed. Axis and conventions are defined and finally, the threat encounter scenario is mathematically modelled.

In **Chapters 5 and 6**, the collision avoidance algorithms are designed and tested.

In **Chapter 7**, Monte Carlo simulations and analysis is performed on the algorithms designed in the preceding chapters.

In **Chapter 8**, a detailed comparison of the algorithms is given. This is coupled with a discussion and interpretation of the test results.

Finally, **Chapter 9** concludes the thesis by uniting all the information produced by the foregoing chapters. It essentially provides a unified perspective for the thesis in its entirety.

Chapter 2: Literature Review

Collision avoidance systems and algorithms are the topic of a significant portion of the research effort in the academic community. All of these efforts have sought to find a solution to sense and avoid or its constituent functions.

Before the advent of UAVs, collision avoidance was researched vigorously under the umbrella of Conflict Detection and Resolution (CD&R) for Air Traffic Control (ATC) [8]. These techniques have been applied to the CA problem for UAVs as sense & avoid and CD&R are effectively the same.

In comprehensive surveys by Kuchar and Yang [8], [9], the authors have stated that CD&R (*and by implication sense and avoid*) can be categorised by the following six design factors:

1. *State Propagation*
2. *State Dimensions*
3. *Conflict Detection*
4. *Conflict Resolution*
5. *Resolution Manoeuvres*
6. *Multiple Conflicts*

Another thorough survey has been performed by Mujumdar and Padhi [10]. In this text, they subdivide the UAV collision avoidance problem into two categories: *Global Path Planning* and *Local Collision Avoidance.*

In what follows, a thematic survey is given in the context of this thesis. It is divided into two subsections based on the knowledge of the surveys mentioned previously. The first section is Collision Detection. These are the algorithms which will satisfy the *Evaluate, Prioritise and Declare* functions described in Chapter 1. The second section is Collision Avoidance. These are the algorithms which will satisfy the *Determine, Command and Execute* functions which are also described in Chapter 1.

The reader should note that only non-cooperative collision avoidance has been reviewed in this thesis.

2.1 Collision Detection Algorithms

Collision detection is the act of determining whether in some future time, a *target* detected becomes a *threat* to the safety of the UAV. This is achieved by state propagation of the UAV and threat's states. Kuchar and Yang [9] have determined that in general, there are three methods to achieve this. These are graphically shown in Figure 7.

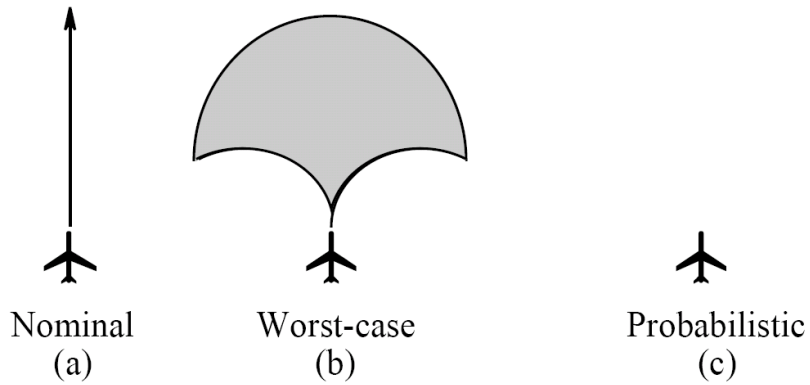

Figure 7: State propagation methods described Kuchar and Yang [9].

The Nominal method disregards any uncertainty in the state information and propagates the velocity along a single trajectory. Its simplicity makes it a very attractive solution.

Alternatively, the Worst-case method determines the range of manoeuvres the threat aircraft can possibly execute and propagate each of these trajectories. Subsequently, each of these trajectories is tested for collision. This method is conservative by its nature and will often result in false alarms.

In the Probabilistic method, uncertainty is modelled as deviations from the nominal trajectory. This method has the potential of producing an optimal

solution, however these methods are generally complicated and it is also difficult to model the probability distribution for the state deviation.

Using this information, the literature has been further subdivided below.

2.1.1 Geometric Methods

Geometric Methods employ the use of an aircraft's position and velocity states to determine geometrically if a collision will occur. They are generally simpler (and relatively easier to grasp) than other methods and it is this property that makes them appealing.

The most popular of these methods is the Collision Cone developed by Chakravarthy and Ghose [11]. This collision detection method has been "rediscovered" throughout the years. The Maneuvering-Board Approach [12], The Velocity Obstacle [13] and Forbidden Velocity Maps [14] are all essentially the same idea as the Collision Cone.

Fundamentally, the Collision Cone is the set of velocities which will result in the collision of two objects. In other words, if A and B are constant velocity objects and A's velocity vector falls within a certain "cone", the two objects will collide. This is shown in Figure 8 below.

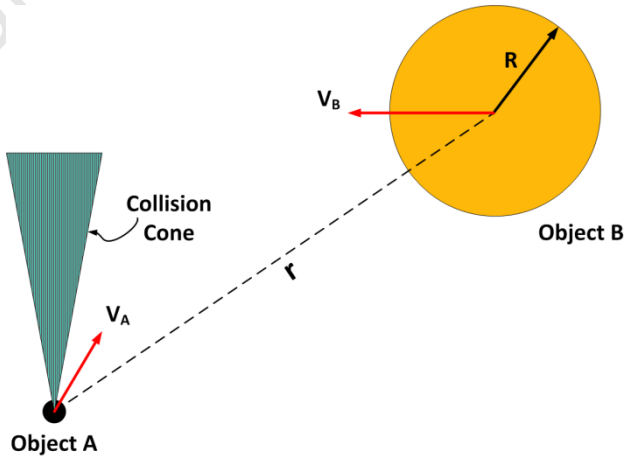

Figure 8: Collision Cone concept. The cone contains all the velocities which will result in a collision between the objects. 'R' is the radius of object B and 'r' is the line of sight. Note that Object A is a point mass in this specific scenario.

The Collision Cone has also been extended to 3D collision detection of spherical stationary objects by Watanabe [15]. This is performed by forming a plane with the relative position (**X**) and velocity (**V**) vectors shown in Figure 9.

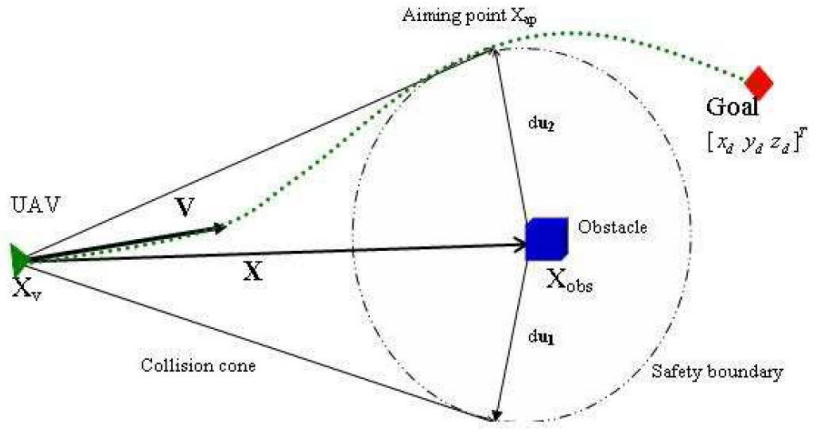

Figure 9: Adapted Collision Cone for 3D collision detection [15].

Subsequently, the Collision Cone was also extended to 3D moving obstacles in the MSc thesis by Melander [16].

The Collision Cone does suffer from a lack of robustness to uncertainty in state information. There have however been attempts at alleviating this using Probability Velocity Obstacles (PVO) [17], [18]. Unfortunately, these methods become more complex and this causes them to lose the attractive simplicity of the original Collision Cone.

2.1.2 Probabilistic Methods

Probabilistic methods address the issue of uncertainty in the threat aircraft motion [9]. However, before the algorithm can be designed, sufficient knowledge of the statistical nature of the airspace is required as this will be used to determine the probability of collision.

Kim, Park & Tahk [19] employed an online Monte Carlo Simulation to calculate the probability of collision between aircraft. A similar method was performed by Lindsten, Nordlund & Gustafsson [20], where the collision risk is continuously calculated at each time step using Monte Carlo Approximations. This process is illustrated graphically in Figure 10.

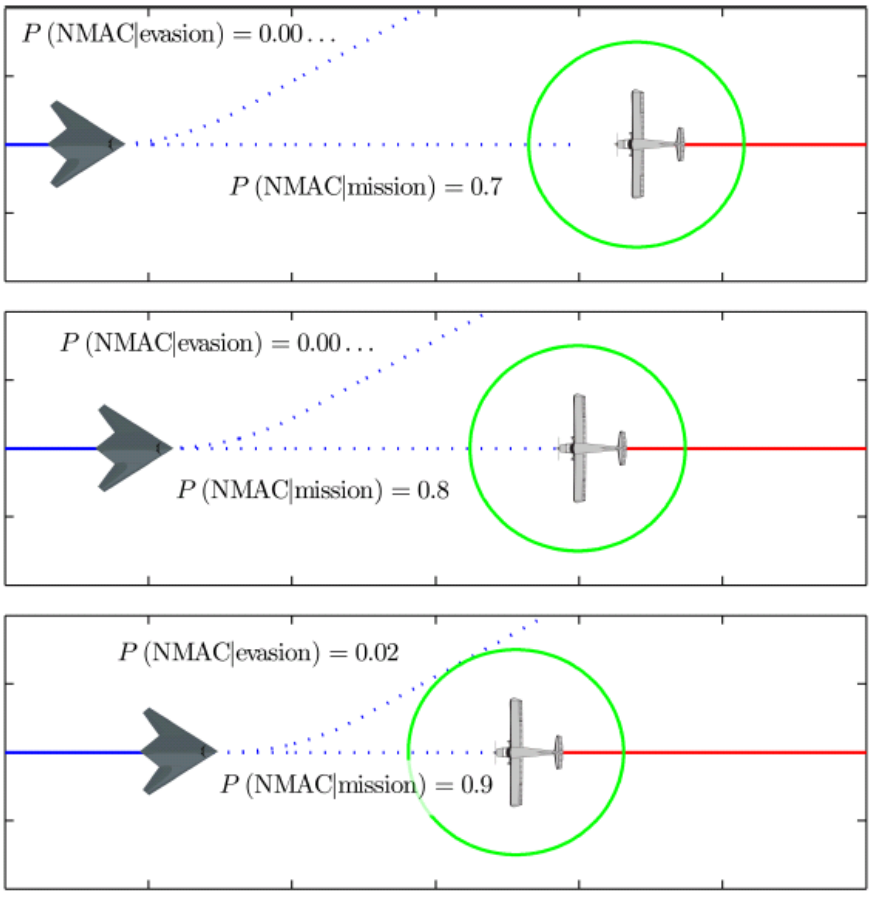

Figure 10: The online risk computation used in [20]. In the last step, the risk is above the threshold and an evasive manoeuvre is triggered.

These methods are interesting as they address the uncertainty aspect associated with the threat motion. Nonetheless, the major shortcoming is that online Monte Carlo Simulation is computationally expensive.

2.2 Collision Avoidance Algorithms

Collision avoidance algorithms are reviewed in a similar fashion to that performed in the survey by Mujumdar and Padhi [10].

2.2.1 Global Path Planning Algorithms

Global Path Planning algorithms have been utilized primarily in the robotics field to determine a collision free continuous trajectory (or path) for an autonomous vehicle [21]. These algorithms use global information to plan a feasible path to a goal.

The most popular of these methods is the deterministic A* search algorithm [22] which employs a grid or graph search to find a path to a goal. This algorithm can also factor in the kinematic constraints of the UAV [23]. However, this method is generally computationally intensive.

There has been a new method developed by Belkhouche [24] called Reactive Path Planning. This method addresses the issue of dynamic obstacles by converting to a “Virtual Plane”. This Virtual Plane maps moving obstacles to look like stationary objects which greatly simplifies the generation of the path. This method has only been applied in 2D, but it is this author’s view that this algorithm could serve as a possible solution to the UAV collision avoidance problem in the near future.

Other methods such as Rapid Exploring Random Trees (RRTs) [25] have also been utilized for autonomous global path planning among obstacles [26]. The primary disadvantage is that these methods are largely open loop and computationally intensive. The reader should note that efforts have been made to address these issues [27] [28].

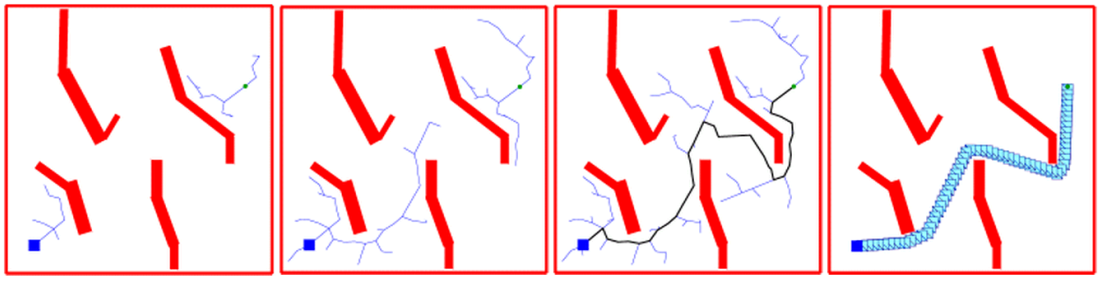

Figure 11: RRT planning a path from an initial point (blue square) to a goal point (green point). Image courtesy of [29].

2.2.2 Local Collision Avoidance Algorithms

Local collision avoidance algorithms are reactive in nature as they only avoid obstacles in their immediate vicinity. These algorithms do not have knowledge of the global environment and rely on local information provided by on-board sensors to determine avoidance manoeuvres.

A significant portion of these algorithms are based on Proportional Navigation (PN) methods [30], [31], [32], [33], [34]. PN is a missile guidance technique where the algorithm commands an acceleration perpendicular to the Line of Sight (LOS) between a missile and a target [35]. In the context of collision avoidance between UAVs, this acceleration can be proportional to the relative velocity vector as in the work of Han [30] or the UAV's velocity as in the thesis of Smith [34]. An example of the work by Han is shown in Figure 12.

These PN methods are broadly successful in simulation as the acceleration commands are able to aggressively guide the UAV to safety. However none of these papers (except Smith [34]) make mention as to how these commands interface to the UAV at an acceleration level. In addition to this, the lateral (perpendicular) acceleration will cause the aircraft to sideslip, which is inefficient aerodynamically and can cause the aircraft to stall.

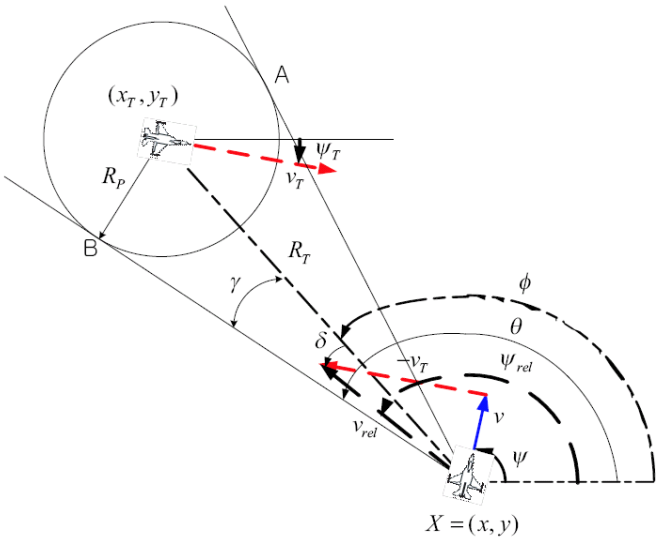

Figure 12: Encounter scenario for PN collision avoidance. The relative velocity vector is guide to the collision avoidance vector XB by a lateral acceleration [30].

Other acceleration methods have also been developed by Mujumdar and Padhi [10] and Watanabe [15]. These have demonstrated success in reactively evading stationary targets in simulation. However, these suffer from the same problems as the PN Methods mentioned above.

Model Predictive Control (MPC) has also been employed in the collision avoidance problem. Shim and Sastry [36] have demonstrated a MPC algorithm to evade collisions between UAVs. This algorithm was successful in flight tests between two rotary wing UAVs. This technology has not been applied to fixed-wing UAVs yet and furthermore, MPC algorithms are generally computationally intensive.

A Geometric approach has also been demonstrated by Goss, Rajvanshi and Subbaro [37]. This method uses a combination of Collision cones and Geometric methods to obtain collision avoidance in 3D. However, this algorithm implements a nonlinear solver, which in the words of the authors' "has a tendency to get stuck" as per page 19 of [37].

2.3 Summary

A summary of the collision avoidance and detection systems is given below as per the discussion of this chapter.

In summation:

- Geometric Collision Detection methods provide the simplest and most well understood methods. These methods are generally well understood and quick to implement but do not factor in uncertainty.

- Probabilistic Collision Detection methods can incorporate uncertainty in their calculations. However, these require one to model the probability distribution of various target motions in the airspace and this data is not readily available. Coupled to this is the fact that the online Monte Carlo simulations required by these algorithms, are computationally expensive.

- Global Path Planning algorithms tend to be computational expensive as well as requiring knowledge of the entire environment to operate optimally in the global sense.

- Local Collision Avoidance algorithms are attractive in that they only require on board sensor information to perform manoeuvres. It has also been demonstrated in the literature that acceleration-based methods have been successful.

In this thesis, the Collision Cone approach [15] will be utilized for collision detection. The primary reasons being that it is computationally efficient, simple to implement and easily extendable to multiple threats.

Acceleration-based collision avoidance methods such as Proportional Navigation are a feasible choice for collision avoidance. This is based on the literature illustrating that acceleration commands can provide the aggressive manoeuvres required for successful reactive collision avoidance. However, the shortcomings inherent to these methods (means to interface to UAV and sideslip, etc.) need to be addressed first.

In the following subsection it will be demonstrated how a recently published, novel acceleration based method of guidance for UAVs can be exploited to address these deficiencies.

2.4 Specific Acceleration Matching Control

The lateral acceleration and acceleration interface issues described previously can be addressed by Specific Acceleration[6] Matching (SAM) Control [38]: A robust, computationally efficient method of control for UAVs. This method has successfully been applied to Aerobatic Flight [39], Variable Stability Flight [40] and Autonomous Take-Off [41]. In what follows, a conceptual overview of SAM Control is provided. This knowledge will form the baseline on which the collision avoidance algorithms and UAV model will be implemented in the forthcoming sections of the thesis. The reader should note the particular mathematics will be dealt with in the forthcoming chapters of the thesis and the purpose of this subsection is merely to provide conceptual understanding.

[6] Specific Accelerations are all the accelerations affecting the aircraft except for gravity.

2.4.1 Axis Systems

The axis systems defined below will form the basis upon which the SAM Control theory is developed and explained. Thus, an insight into them is vital for the reader.

The Inertial Axis System [42] provides a fixed reference frame whereby the aircraft motion can be described. This right hand orthogonal axis is fixed to a flat and non-rotating Earth, with its origin at some arbitrary reference point [43]. The North-East-Down directions are shown in Figure 13.

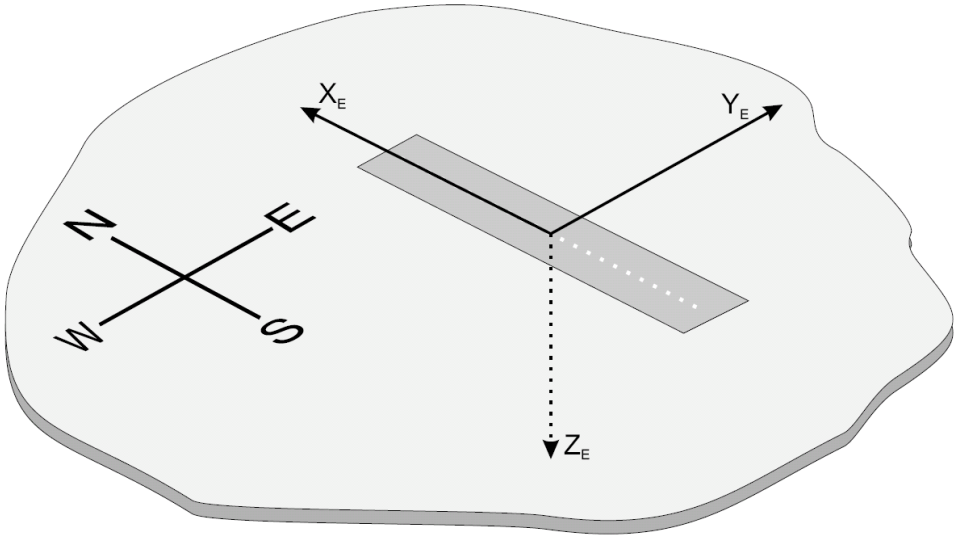

Figure 13: The North, East and Down directions corresponding to the X, Y and Z axes respectively [40].

The next important axis system is the Wind Axis System [43]. The orientation of the wind axis is defined with the X_W axis always pointing in the direction of the total velocity vector ( ), the Z_W axis is perpendicular to the X_W axis and is on the aircraft's plane of symmetry, and the Y_W axis is such that X_W-Y_W-Z_W forms a right handed system [8]. Also, observe that the unit vectors defining the wind axis are $\boldsymbol{i_W}$, $\boldsymbol{j_W}$ and $\boldsymbol{k_W}$ for the X_W, Y_W and Z_W axes respectively and the origin is the Centre of Gravity of the aircraft.

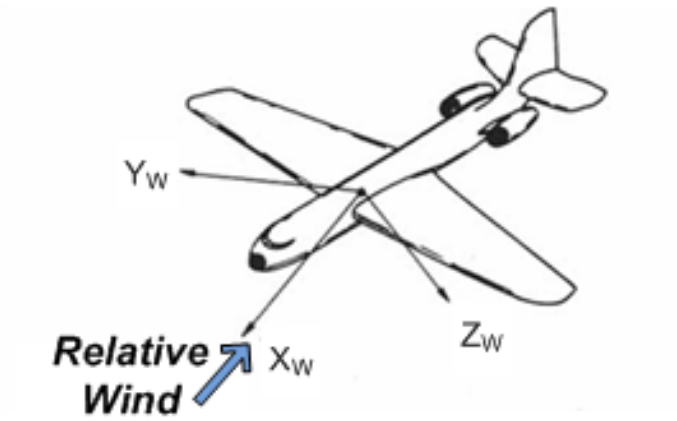

Figure 14: Wind Axis System definition [43].

2.4.2 SAM Control Theory

It is shown by Peddle [38], that by the principle of timescale separation[7] the aircraft dynamics can be split into two subsections: The rigid body kinetics and the point mass kinematics. The rigid body kinetics contains all the aircraft specific force and moment dynamics (*Inner Loop*) and the point mass kinematics (*Outer Loop*) contain the position and velocity dynamics. This is graphically shown in Figure 15.

[7] Timescale separation is when dynamic behaviour in a system is much faster (or slower) that they can be ignored.

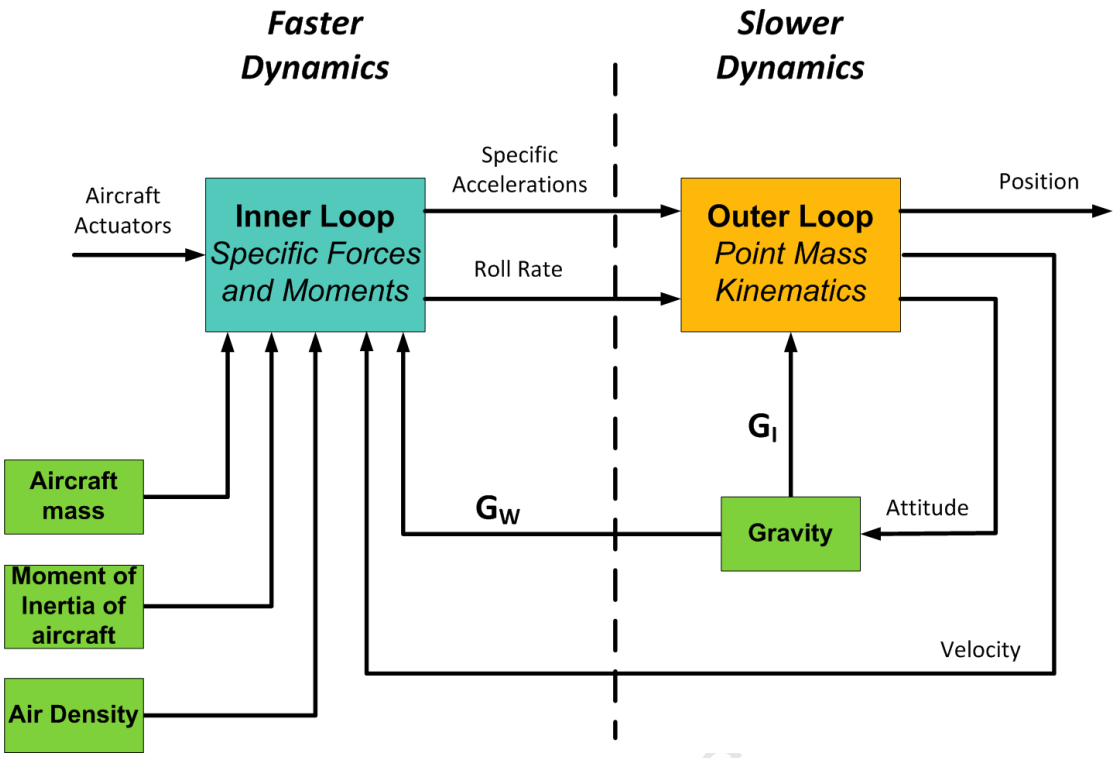

Figure 15: Total aircraft model is split by timescale separation into its fast kinetics and slow kinematics models. $\mathbf{G_I}$ and $\mathbf{G_W}$ are the Gravitational vectors coordinated into Inertial and Wind axes respectively.

The reader should note that the Inner Loop specific forces and moments are defined in the Wind Axis System.

It is also proven by Peddle, that if high-bandwidth inner loop Specific Acceleration controllers can be designed, then from a guidance perspective the aircraft reduces to point mass under the influence of "Virtual Actuators". Peddle has declared that a time separation factor of five times is generally sufficient [38].

These Virtual Actuators are defined as:

- **Axial Specific Acceleration Controller (A_W) or ASA:** This is the acceleration along the X_W axis. Its underlying aircraft actuator is the thrust (or throttle).

- **Normal Specific Acceleration Controller (C_W) or NSA:** This is the acceleration along the Z_W axis. Its underlying aircraft actuator is the elevator.
- **Lateral Specific Acceleration Controller (B_W) or LSA:** This is the acceleration along the Y_W axis. Its underlying aircraft actuator is the rudder.
- **Roll Rate Controller (P_W):** This is the roll rate about the X_W axis. Its underlying aircraft actuator is the aileron.

Consequently, if these virtual actuators are in place, the outer loop guidance controllers can view the aircraft as an instantly commandable specific acceleration vector [38]. This greatly simplifies the design of the guidance controllers as they do not have to factor in the aircraft kinetics. This concept is shown graphically in Figure 16.

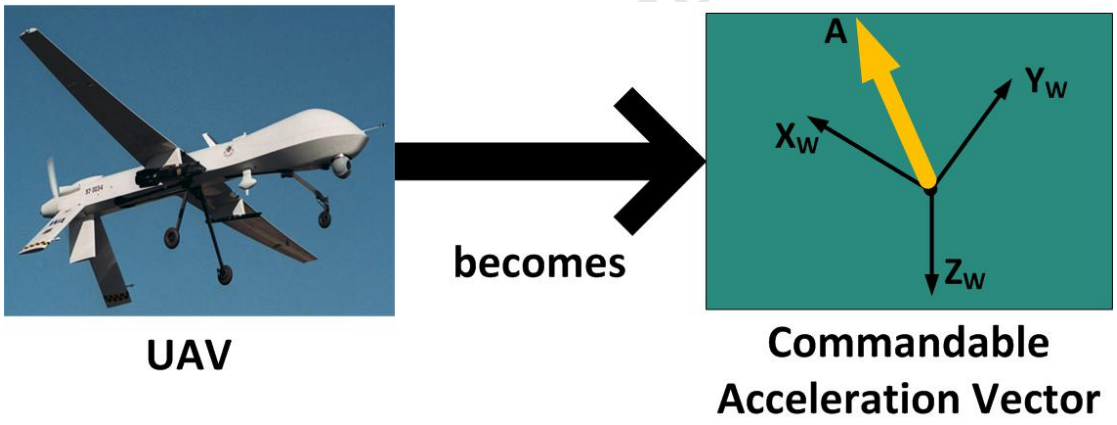

Figure 16: The complicated UAV model is reduced to a commandable acceleration vector by use of Virtual Actuators.

2.4.3 SAM Control Guidance Architecture

The SAM Architecture greatly simplifies the guidance problem for UAVs. The detailed derivation of these guidance controllers has been illustrated by Peddle [38] but for the sake of completeness, an outline of each controller and algorithm is provided below.

With the inner loop (Virtual Actuators) previously described, the UAV can be viewed as a point mass under the influence of specific accelerations and a time-invariant gravity vector fixed to the inertial frame. The reader should note that the guidance dynamics will be explained in greater detail in Chapter 4. This chapter serves more as a conceptual overview to assist the reader in grasping the later chapters.

An overview of the general guidance architecture developed by Peddle [38] is shown:

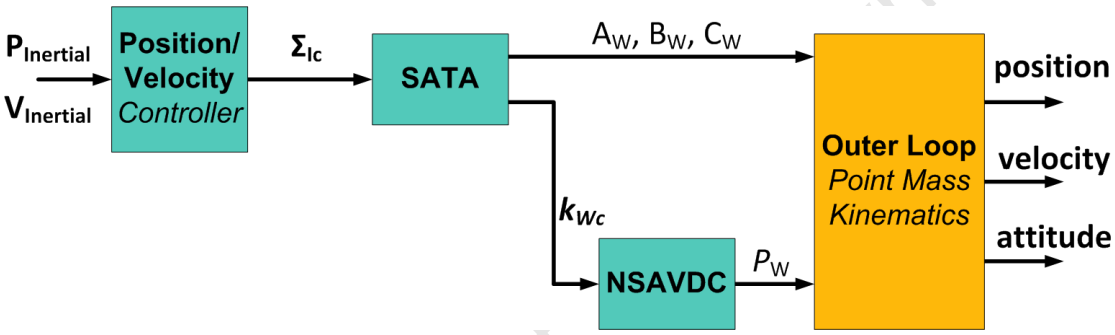

Figure 17: General SAM Guidance Architecture.

In this architecture, the outer most control loop is either the position and/or velocity control algorithm. These are generally implemented as standard proportional controllers as the natural free integrator in the dynamics will ensure zero steady state error [38]. These controllers are generally sent a reference command by a trajectory generator or path planner and using this, they command the required specific acceleration with respect to inertial space [39], [41].

The Specific Acceleration Transformation Algorithm or SATA then matches the required acceleration in the inertial frame ($\boldsymbol{\Sigma_{Ic}}$) to an equivalent specific acceleration command in the wind axis system ($\boldsymbol{\Sigma_{Wc}}$). It does this by splitting the specific acceleration vector into its constituent components and

by rolling the wind axis to an appropriate attitude [39]. These constituent components are A_W, B_W and C_W.

However, Peddle [38] has stated that if coordinate flight[8] is desired, then B_W should not be used by the guidance controllers. Furthermore, Gaum [39] has illustrated that the Axial Specific Acceleration, A_W, should also not be used for guidance. This is due to the bandwidth limited nature of its underlying actuator (the engine) which causes it to break the time scale separation assumption.

In light of the aforementioned information, the only virtual actuator employed by the SATA should be the Normal Specific Acceleration (NSA) which can then be rolled to the require direction. Thus, the SATA can only command an acceleration vector lying in a plane spanned by the unit vectors $\boldsymbol{j_W}$ *and* $\boldsymbol{k_W}$ and any acceleration component lying in the axial direction is to be removed**.** The aircraft, thus reduces to an NSA which can be rotated about $\boldsymbol{i_W}$**.** The concept is illustrated in Figure 18.

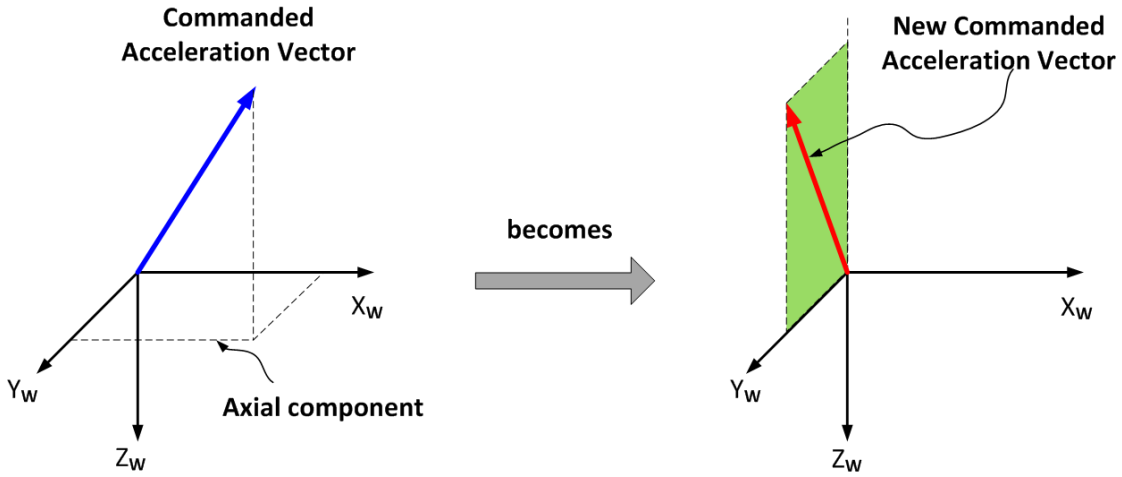

Figure 18: Conceptual diagram of the SATA. The axial component is removed to obtain a specific acceleration lying in the Y_W-Z_W plane.

[8] Coordinated flight is flying with zero sideslip.

The next task is to take the commanded NSA and rotate it to the desired direction. The **Normal Specific Acceleration Direction Controller (NSAVDC)** performs this by taking the commanded direction $\boldsymbol{k_{Wc}}$ and the current $\boldsymbol{k_W}$ and calculating the error angle (φ) between them [38]. It then employs a proportional controller to drive this error angle to zero by commanding a roll rate (P_W) about $\boldsymbol{i_W}$. The reader should note that this angle is not necessarily equivalent to the aircraft bank angle. A view of the Y_W-Z_W plane is shown in Figure 19 presenting the concept.

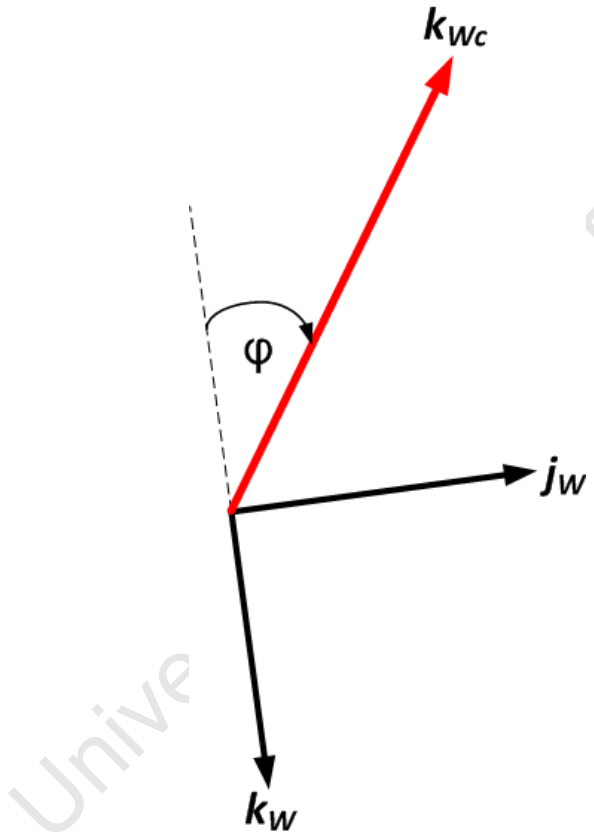

Figure 19: View from the Y_W-Z_W plane illustrating the error angle (φ). This angle is driven to zero by the NSAVDC with a proportional roll rate.

Now that a conceptual review of SAM Control has been provided, the reader will be more equipped to understand the succeeding chapters. It has been shown how SAM Control theory can be used as an effective method for guidance of UAVs. In addition to this it also addresses the issues of previous acceleration-based methods for collision avoidance. Thus, this theory will be exploited in the design of the collision avoidance system for this thesis.

2.5 Aircraft Limitations

In order to adequately design guidance and control algorithms for aircraft, a measure of insight into aircraft structural and manoeuvrability limitations needs to be attained. This subsection provides the reader with the means to understand why certain design decisions were made during the later chapters of this thesis.

2.5.1 Load factors definition

Load factors are defined as the ratio of the lift of an aircraft to its weight [44]. The forces acting on an aircraft during a constant altitude, coordinated turn are depicted in Figure 20. During this turn the aircraft will experience a total load on its structure of two times its gross weight.

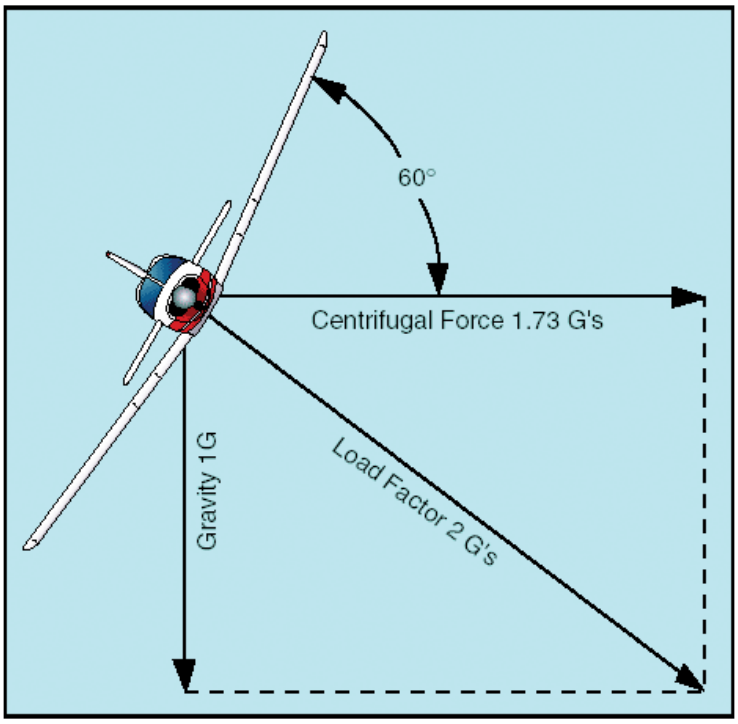

Figure 20: Image of the centrifugal and gravitational forces acting on an aircraft during a coordinated turn [45].

Load factor is calculated by considering that during a coordinated turn, to maintain altitude the lift force must equal the gravitational force [46]. This is shown by equation (2.1). Note that is the bank angle, W is the weight and L is the lift.

(2.1)

Then by some algebraic manipulation, the load factor (is shown below:

——— (2.2)

A plot of the load factor as a function of bank angle is shown by Figure 21.

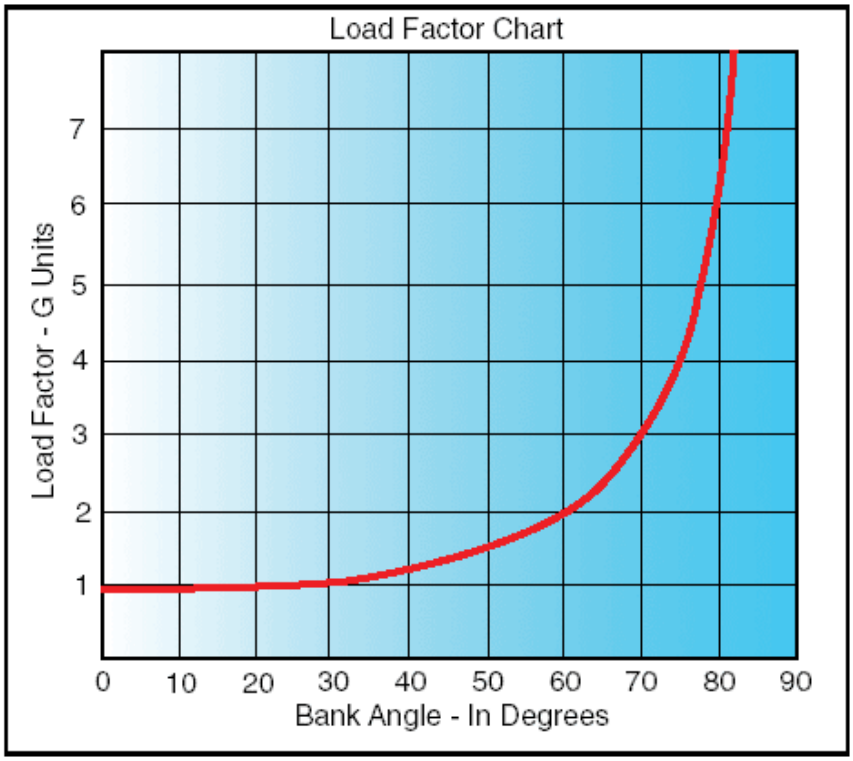

Figure 21: Load factor as a function of bank angle [45]. Note that the load factor increases dramatically as the bank angle approaches 90°.

It is evident that for bank angles lower than 40°, the extra load factor induced to maintain altitude is negligible. Comparatively, bank angles greater than 40° require large load factors to maintain straight and level flight.

This section provided a brief introduction into the effects of load factor on manoeuvrability. In the following section, the effects load factor have on airspeed will be discussed.

2.5.2 Load factors and airspeed

Excessive load factors will increase stall speed of an aircraft [47]. In fact, one can think of the load factors as increasing the stall speed. In the context of UAVs, safety is of paramount importance and stalling the aircraft could result in catastrophic results. Besides the fact that most UAVs carry multi-million dollar payloads, stalling the aircraft could potentially result in damage to buildings as well as injury to people.

Lift force, defined by equation (2.3), is created primarily by the wing of an aircraft. This lift force is proportional to a coefficient of lift (, which is specific for each type of wing, the air density ( ), wing area (*S*) and the true airspeed [44].

– (2.3)

A diagram of a typical lift-drag (L-D) curve is provided by Basson [48] and is illustrated in Figure 22.

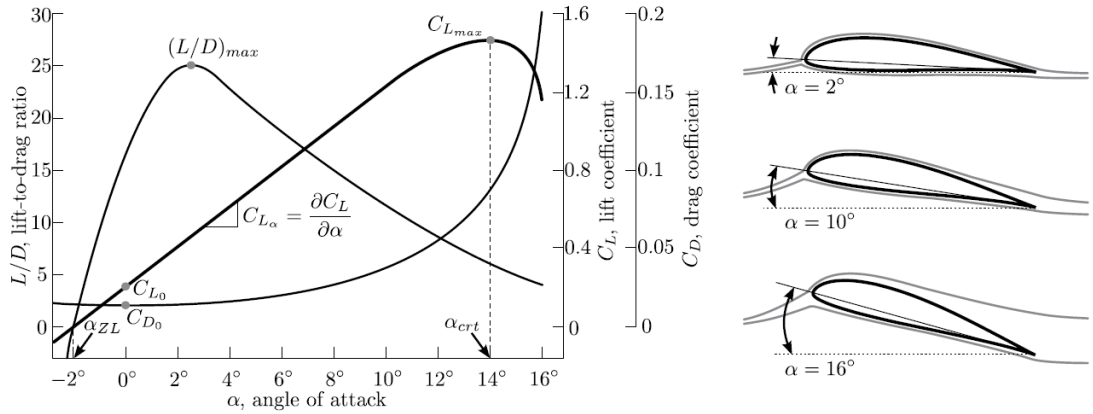

Figure 22: Typical L-D curve for a conventional wing. Notice that at an angle of attack of 14 degrees the lift, this wing stalls.

From this diagram, it is evident that there is generally a maximum amount of lift that the wing can provide the aircraft. As the wing approaches the critical angle of attack, the wing loses this lift dramatically [47]. Additionally, lift is also generated by the airspeed as indicated by equation (2.3). The minimum airspeed at which the aircraft can maintain the weight of the aircraft is known as the “stall speed” [44].

The reader should note that the topic of stall is a vast one and has only been mentioned in this section. For a more in depth understanding the, the reader is encouraged to consult the thesis by Basson [48].

Now, consider an aircraft in a coordinated turn. The lift required to keep the aircraft at a constant altitude is given by [44]:

(2.4)

Thus, at a higher load factor, more lift is required. Considering the aforementioned critical angle of attack, there is a limit to this lift force before the aircraft starts to stall. From the literature [49] the accelerated stall speed is given by,

–

(2.5)

Where is the stall speed at straight and level flight (load factor of one) and is the accelerated stall speed induced by load factor.

To graphically illustrate this, most have aircraft a v-n curve [47]. This is a curve depicting the aircraft limit load factor as a function of airspeed. A typical example is shown in Figure 23.

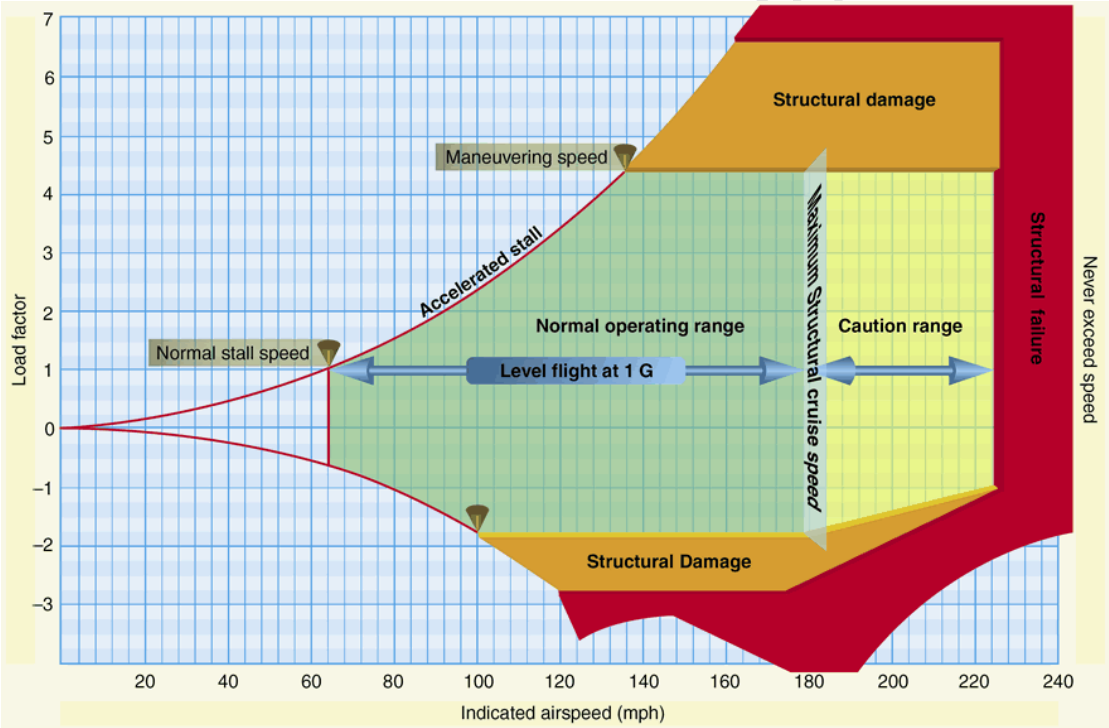

Figure 23: Typical v-n diagram for a general aircraft [50].

It is evident that this diagram gives the capability envelope of the aircraft. From this diagram, it is shown that as the forward airspeed increases, so does the maximum load factor. As discussed previously, this is due to the maximum lift capability increasing proportionally with an increase in airspeed. However, as depicted from the figure, there is a maximum load factor. This is

due to the aircraft approaching the critical angle of attack, which will result in stall.

A similar situation exists for negative lift. The reader will note that the graph is *not* symmetrical about the airspeed axis. This is due to the fact that most aircraft are not designed to experience large negative load factors or to dive rapidly. This can be attributed to the design of their wings [47].

Also, from this diagram it can be seen that any load factor above 4.4 will result in structural damage to the aircraft. This is illustrated by the orange region on the graph, which should be avoided when flying the aircraft.

Additionally, the intersection of the positive limit load factor and the line of maximum lift capability is of particular importance. The airspeed at this point can be seen as the minimum airspeed at which the limit load can be created. Any airspeed beyond this threshold will indeed also result in damage to the aircraft. This is usually referred to as the “manoeuvring speed” in the literature [47] and the aircraft is generally operated below this speed. A comparable situation exists for the negative limit load factor.

The limit airspeed (*Never Exceed Speed*) or [51] depicted by the red line, is very important when operating an aircraft. This speed will cause the aircraft’s wings to fail structurally if exceeded.

Figure 23 provides an overview of the operational envelope of the aircraft. It provides both the maximum and minimum load factors, as well as the maximum airspeeds. The pilot or flight control engineer must thus be particularly vigilant of these limitations when operating the aircraft.

2.5.3 Summary

The limitations on aircraft performance have been briefly explained in this subsection. In summation, increasing the load factor has two main effects. The first effect is structural damage: Increasing the load factor increases the stress on the aircraft structure. The second effect is stall speed: Increasing the load factor increases the stall speed.

Additionally, the v-n curve was introduced by means of Figure 23. This diagram graphically depicted the aircraft operational envelope of a general aircraft. It is used as reference as a guideline for pilots and flight control engineers to determine the safe airspeeds and load factors which can be experienced by the aircraft. From this diagram, the *Never Exceed Speed* has been shown graphically. This is an important factor that limits aircraft performance as well as safe operation.

These aircraft limitations will be a critical deciding factor in the design of the CAS.

Chapter 3: Methodology

This chapter describes the research process employed in this thesis project. The aim of this chapter is to give the reader insight into *how* this thesis would approach the collision avoidance problem for UAVs. The procedures followed, the data synthesis and data analysis methods are altogether described below.

3.1 Overview of the Model Based Design Approach for Research

The Model based design approach (MBDA) [52] is an effective mathematical method for designing complex systems in the areas of Control, Signal Processing, Power and Communication. It has seen great success in the Aerospace and Automotive industries as illustrated in [53], [54] and [55].

Fundamentally, the MBDA consists of the five phases shown below. The approach is visualized in Figure 24. These phases are listed as:

1. ***Research and Requirements*** – The process starts by researching the problem and evaluating previous solutions. At the same time, a list of requirements is recorded based on the research results.
2. ***Modelling*** – The "Plant" (i.e. the environment to be investigated) [52] is modelled by means of first principle mathematics or empirical data. The Plant comprises the environment (and all its sub-components) which the System[9] will interact with. The "Plant Model" is produced in this phase.
3. ***Algorithm Design*** –The Plant Model's dynamic characteristics are investigated and applicable algorithms are designed. These algorithms are designed based on the outcome of the Research and Requirements phase.
4. ***Simulation*** – The algorithms are tested against a simulated Plant. The results are then thoroughly analysed and are measured against

[9] In this context, "the System" can be seen as a collection of algorithms functioning together.

any performance metrics. Phase 3 can be iterated to optimise performance if desired. Additionally, Phase 2 can be repeated to produce a higher fidelity model.

5. ***Deployment*** – Once the design requirements are satisified, the algorithms are ported to embedded hardware or integrated with other algorithms.

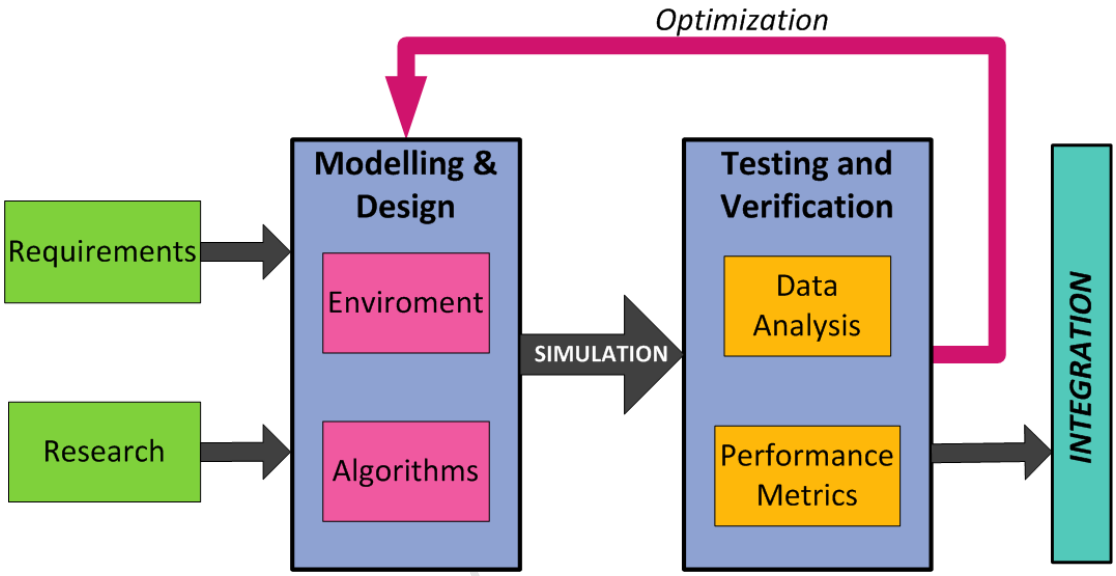

Figure 24: Model Based Design Approach (MBDA)

The MBDA is illustrated graphically in Figure 24. The requirements and research are seen as the inputs to the entire process. Using these inputs, the modelling of the environment (the Plant) is performed. Additionally, the algorithms are designed based on analysis of the Plant and inputs from the requirements and research performed. Following that the Plant and algorithms are simulated. The data produced by these simulations are then analysed and tested. If performance metrics are not being met adequately, the algorithms or Plant are refined. Finally, once the designer deems the performance as adequate, the algorithms are integrated.

An overview of the design methodology has been presented. The subsections that follow explain the application of this methodology in the context of this thesis.

3.2 Requirements and Research for the CAS

The requirements and research can be considered as the inputs to the process. In the context of this thesis, the requirements were contributed by the SAA Workshop [4] and the NATO Report [6]. Factoring in these requirements, the outcomes of this thesis were presented in subsection 1.3.

The extensive Literature Review performed in Chapter 2 is the 'Research' input to the process. It provided a necessary critical evaluation and investigation into the gaps in the current state of the art knowledge. The information gathered from this chapter will be used as a platform to launch the Modelling and Algorithm Design in Chapter 4.

3.3 Modelling, Design and Simulation for the CAS

Modelling and simulation form a critical part of MBDA. Firstly, the Plant for this thesis is defined. In this context, the Plant is the Collision Encounter (defined in Chapter 1). In a sense, we can consider everything that the CAS interacts with as forming part of this environment. Figure 25 gives a graphical interpretation of the concept. Here we can see that the CAS commands (inputs) are affecting the environment. This will in turn create corresponding states (outputs) for the environment which are fed back to the CAS.

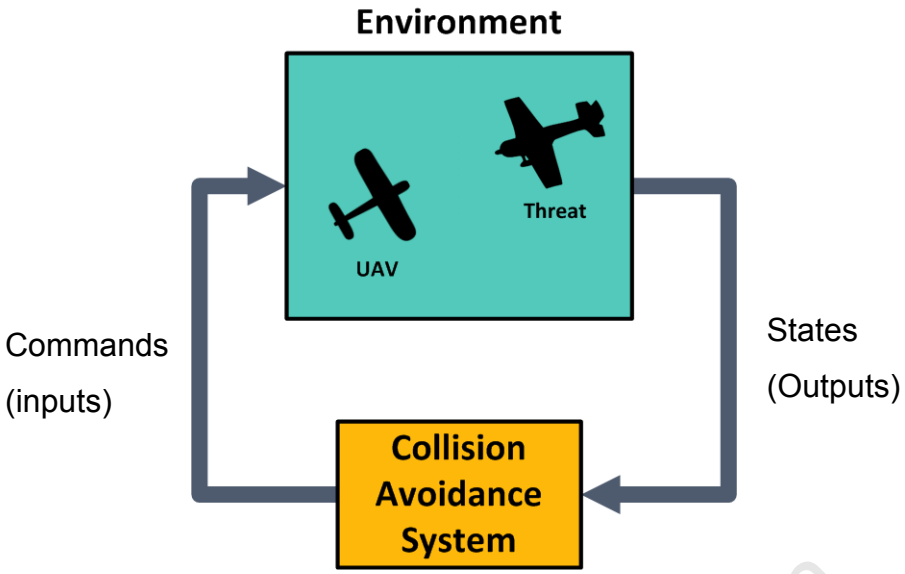

Figure 25: CAS interacting with the Environment.

Thus, in order to model and simulate the environment correctly, the UAV and Threat will need to be mathematically modelled. This is described in the Modelling and Simulation chapter (Chapter 4).

Using these mathematical models, the collision avoidance algorithms are designed. The algorithms are based on the research performed in section 3.3. To verify the designs, extensive simulations are performed.

3.4 Testing and Verification of the CAS

The collision avoidance algorithms are tested in simulation by setting up encounter scenarios. These are performed in a deterministic fashion, where specific scenarios are created one at a time. The simulation data is captured to a hard drive for post-processing and analysis.

In addition to the deterministic simulations, Monte Carlo Simulations [56] are also executed. One of the benefits of Monte Carlo Simulation is it allows complex systems and environments to be tested under random inputs. For the context of this thesis, the encounter scenario (and by implication the threat

states), can be considered as the input. Hence, many encounter scenarios are created where the threat velocity and position are varied randomly. In essence, this provides a measure to the robustness of the CAS. The Monte Carlo Simulation concept is illustrated in Figure 26 below.

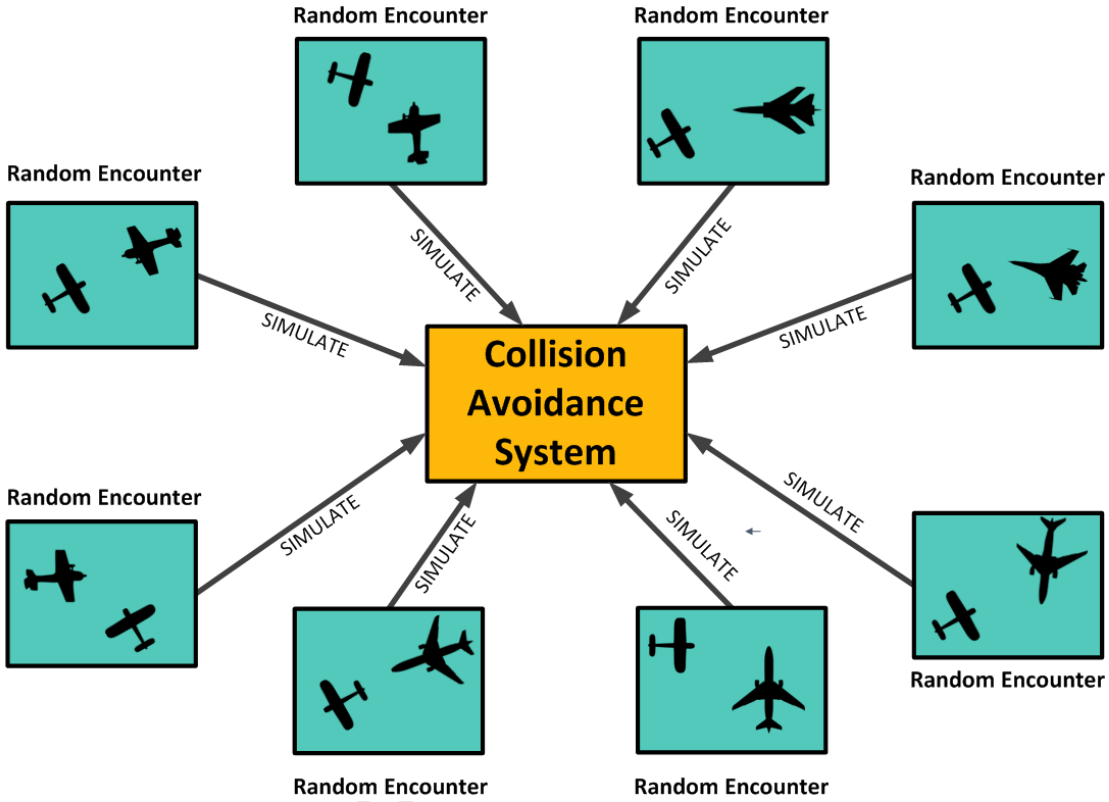

Figure 26: Concept of Monte Carlo Simulation. Numerous threat encounters are generated randomly and then simulated to test the collision avoidance system.

The performance of the collision avoidance system will be measured by the following metrics:

- Collision evaded or not for various Times-To-Impact (TTI) - This will be the paramount measure of the CAS performance.
- Time to Safety (TTS) – The time taken for the CAS to guide the UAV to safety.
- Control Effort – The CAS should not command excessive manoeuvres as this can cause damage to the UAV. For this CAS, bank angle and acceleration commands will be monitored.

- Probability of Near Mid-Air Collision or P(NMAC) – This was defined previously in section 1.3. This metric is applicable for Monte Carlo simulations only. The Monte Carlo simulations provide the data needed to obtain this performance measure.

For deterministic simulations of the CAS, two types of encounter scenarios will be tested as shown in Figure 27. These are:

- **S0** – This is the head on collision where the threat flies in a southerly direction at a bearing of zero degrees from the UAV.
- **S1** – This is where the threat flies due west as a bearing of 45° from the UAV.

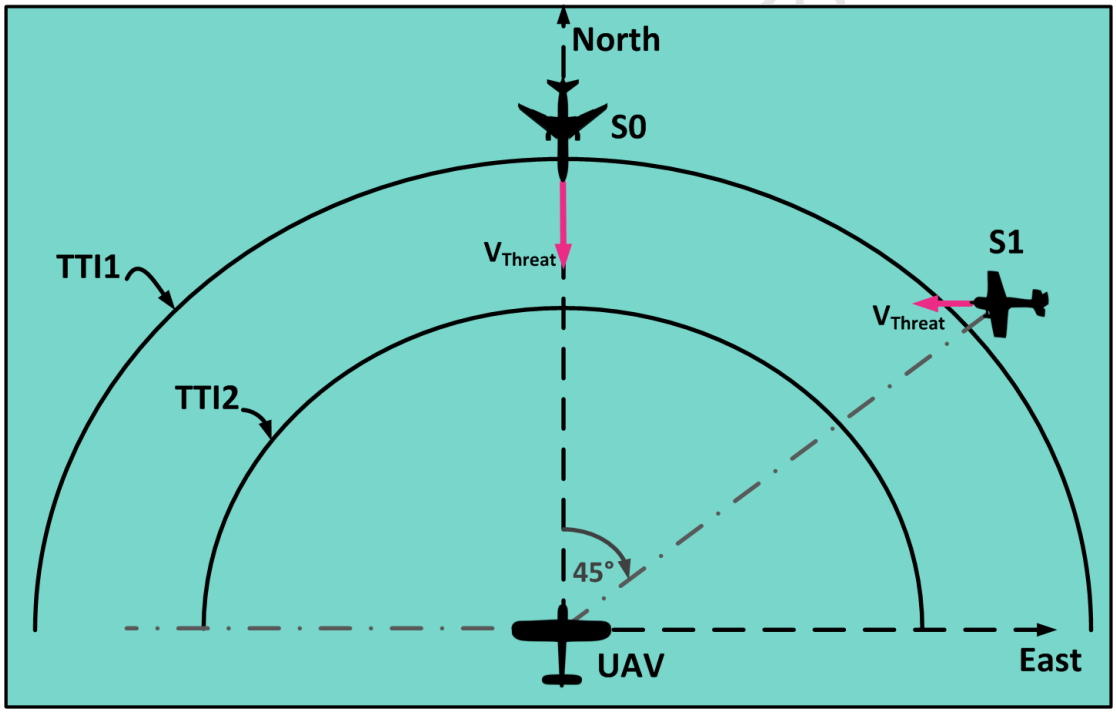

Figure 27: The two types of encounter scenario to be deterministically simulated. The diagram also shows the two TTI test cases.

Additionally, each encounter scenario will also be tested at two TTI values. These are illustrated in Figure 27 and are defined:

- TTI1 = 20s
- TTI2 = 10s

The reader should note that in the simulation, the UAV will start out flying due North at the origin of the NED axis.

The reader is reminded that in Chapter 1 it was also shown that most NMACs occur under VFR flight conditions. Thus, the threats will have a maximum airspeed of 150 m/s as this matches maximum cruise speeds of general traffic in VFR Airspace [57].

The data for each deterministic simulation will be stored in a Matlab data struct and saved to a .mat file[10] for further analysis. Table 1 below illustrates an example entry in the structure of this file. As can be seen, the first simulation was a head on encounter, which had a TTI of 20 s, TTS of 11 s, maximum bank angle of 0.1 rad and maximum NSA of 9.9 m/s^2. Most importantly the binary value of the NMAC field indicates that an NMAC did not occur and thus, the collision was avoided.

Table 1: Data structure of .mat file

Simulation Number	**Encounter Type (S-type)**	**TTI (s)**	**NMAC**	**TTS (s)**	**Max Bank Angle (rad)**	**Max NSA (m/s^2)**
1	0	20	0	11	0.1	9.9
2	1	15	1	N/A	1.51	15.37
3	0	12	0	10.5	1.51	14.04

For the Monte Carlo Simulations, one thousand random encounters for each particular TTI value are generated. The particular TTI values start from a TTI of 25s, to a value of 6s. The Monte Carlo Simulations employ the same data

[10] .Mat files are binary Matlab format files used to store workspace variables.

structure as Table 1 but only the NMAC, TTI and TTS fields are employed in the analysis.

The deterministic simulation data is analysed on a per-simulation basis. The paramount metric considered is if an NMAC has occurred. Secondly, the TTS is also considered for the particular TTI as this will give an indication on how much time the UAV has to account for any latency, or target motion. The control effort metrics are of secondary importance. However, they are monitored, as excessive control effort can cause damage to the UAV as described in Section 2.5.

The Monte Carlo simulation data is analysed by calculating the Probability of Near Mid-Air Collision or P(NMAC). This is performed by taking the total number of collision for a particular TTI, and dividing this by the number of simulations executed. This is shown in equation (3.1).

(3.1)

This calculation is repeated for each TTI for which the Monte Carlo Simulations are executed. A plot of TTI and P(NMAC) will reveal any correlation between the two variables. In addition statistical data will also be obtained for TTS.

Although the full Model-Based Design Approach contains an integration section, this was not implemented in this thesis. The final design should however factor in the idea that the CAS will be integrated into a SAS at a later stage.

Chapter 4: Modelling and Simulation

This chapter describes the modelling and simulation used in the development of the collision avoidance system (CAS). As per Chapter 3, this is a fundamental step in the MBDA and is crucial to adequately design the collision avoidance algorithms. The chapter is divided into three main sections: firstly, the overall concept of the collision encounter to be simulated is discussed; secondly, the kinematic model of the threat is described and finally, the UAV kinematic model is provided.

4.1 Collision Encounter Concept

As described in Chapter 3, in order to design the CAS, we need to adequately model the environment in which it will operate. In this context, the environment is considered to be the collision encounter between the UAV and threat aircraft. Conceptually, this is shown below in Figure 28.

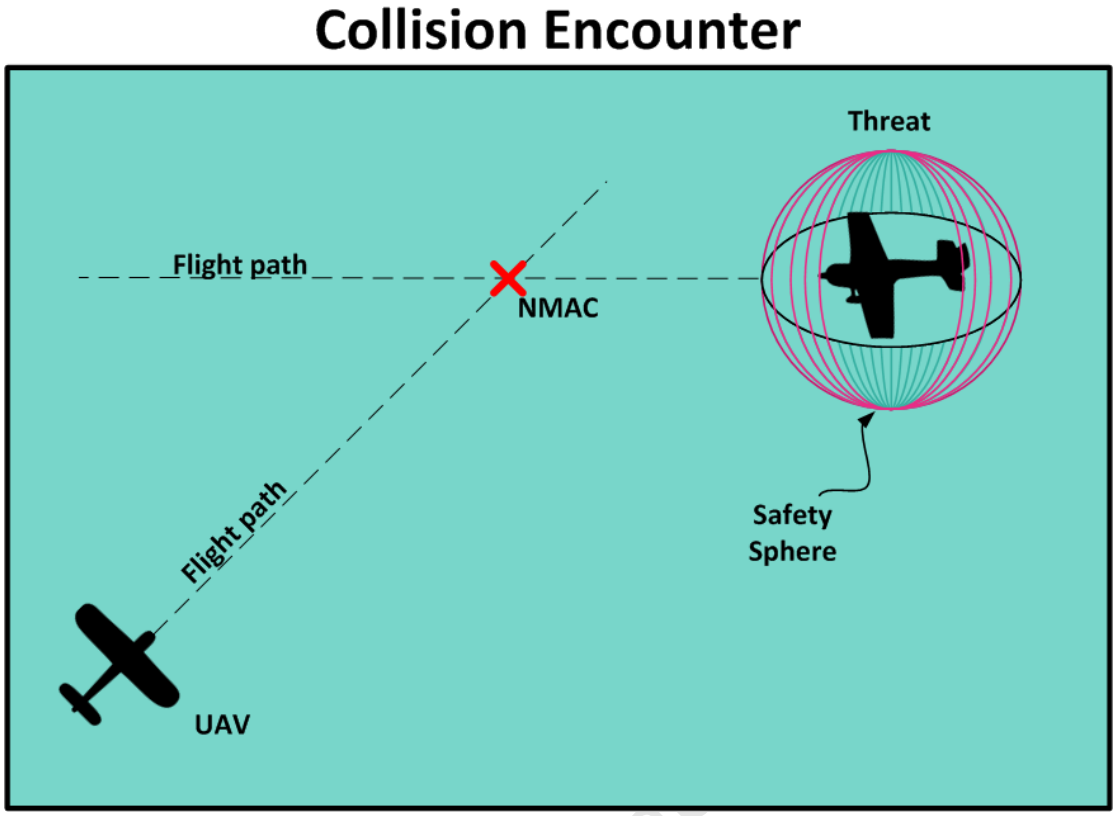

Figure 28: Collision Encounter simulation. The UAV and Threat are following some arbitrary initial flight path which will result in a Near Mid-Air Collision (NMAC).

The model simulates the scenario where the UAV and threat are flying in inertial space along arbitrary flight paths which will result in a Near Mid-Air Collision (NMAC).

The reader should note that there are proprietary collision avoidance simulations in existence. Two of these are SAFEST and CASSATT [4], which are high fidelity models developed from years of NMAC data in the US National Airspace System [58]. Unfortunately, the models and the data they are derived from are proprietary. This fact necessitated a collision avoidance simulation to be developed from the ground up in this thesis.

4.2 Threat Kinematic Model

The threat is a critical part of the collision encounter described previously. Considering the discussions in Chapter 3, a sufficient threat model is critical for thorough development and testing of the CAS.

For this thesis, the threat is modelled as a 3D point mass moving in inertial space. As stated in Chapter 1, we have assumed a constant velocity model. However, in reality, a truly constant velocity is unrealisable due to unmodelled effects such as turbulence, etc. Thus, a “nearly” constant velocity model described by Li and Jilkov [59] was implemented as an attempt to factor in these discrepancies.

The idea behind this model is that for constant velocity, acceleration equals zero. But to factor in the unpredictable effects, a white acceleration noise was added to each acceleration component. This is graphically illustrated in Figure 29.

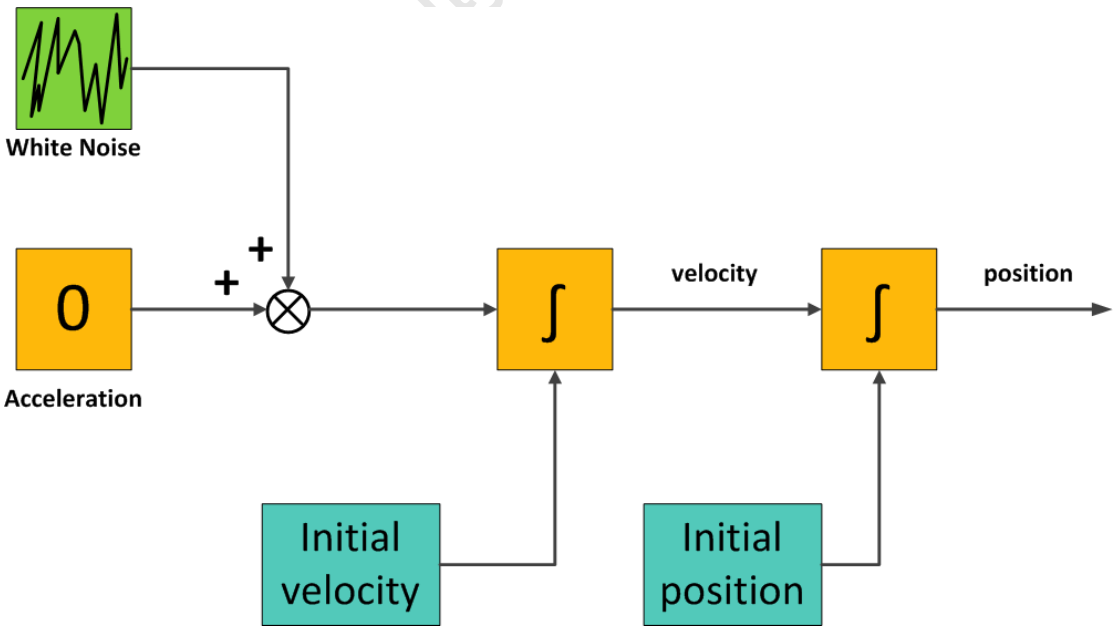

Figure 29: Nearly constant velocity threat model. Each component (X-Y-Z) or (N-E-D) would require this model.

The state equations describing the dynamics are thus:

(4.1)

With:

(4.2)

(4.3)

(4.4)

(4.5)

The reader should note that the model has been adapted from [59] to include an acceleration noise term in the z-component as well.

Finally, the threat is also given a volume to match the requirements described in Chapter 1. However, the following differences between [4] should be noted:

- The collision volume is around the threat and *not* the UAV.
- The collision volume is a sphere and not a cylinder.

The first point is justified by the argument that the collision avoidance problem is commutative[11]. *In other words, a collision between a point mass UAV and a threat with a volume is equivalent to a collision between a UAV with a volume and a point mass threat.*

The second point arises because it is envisaged that it will make the design of the Collision avoidance system simpler. It can also be argued further, that by selecting the threat to be a sphere as opposed to a cylinder provides a more conservative collision avoidance system, which is desirable.

Thus, the threat is modelled as a point mass surrounded by a safety sphere of radius 150 m. The radius is derived from the 500 feet collision volume requirement described in [4].

4.3 UAV Model

The UAV Model (as introduced in Section 2.4), comprises multiple parts. These parts of the model for the CAS are described in the subsections below.

4.3.1 Attitude Definition

Numerous attitude representations are available to represent the attitude of a body in an inertial frame [38]. However, it has been shown by Gaum [39] that the Euler Angle Attitude System (, ,) provides a simple, yet accurate solution. This is pictorially represented by Figure 30 More specifically, this simulation will use the Euler 3-2-1 system: the aircraft's orientation relative to the earth by assuming initial orientation is aligned with the Earth axis, then rotating through the , then and finally angle [41].

[11] The Commutative Property is when the order of an operation does not matter. For example: A + B = B + A.

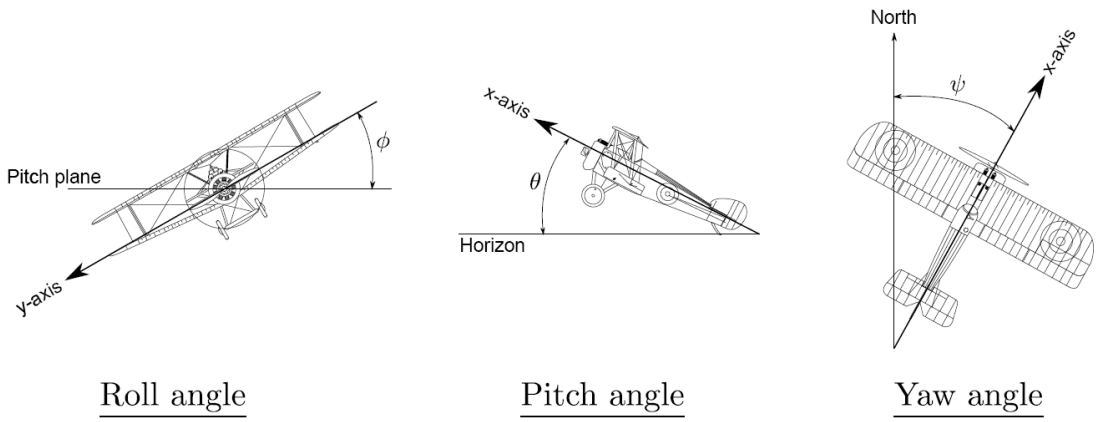

Figure 30: Euler angle attitude system [41].

The reader should note that this attitude system does suffer from a singularity at 90° pitch. But in spite of this, the system is still the most viable as the UAV pitching to 90° is a highly unlikely event in normal flight.

These, angles can be represented in a Direction Cosine Matrix (DCM) derived in [39]. The concept of the DCM and other rotation methods is discussed in Appendix A.1.1.

(4.6)

Peddle [38] also shows that the unit vectors for the Wind-Axis can be calculated from this DCM. The resulting equations are:

(4.7)

(4.8)

(4.9)

4.3.2 Mathematical Model

In light of subsection 2.4, for the purpose of guidance, the UAV can be considered as a point mass under the effect of specific acceleration “Virtual Actuators”. The limitations on the ASA and LSA discussed in subsection 2.4.3 are also to be factored in. Accordingly, the kinematic model of the UAV is illustrated graphically to be that of Figure 31.

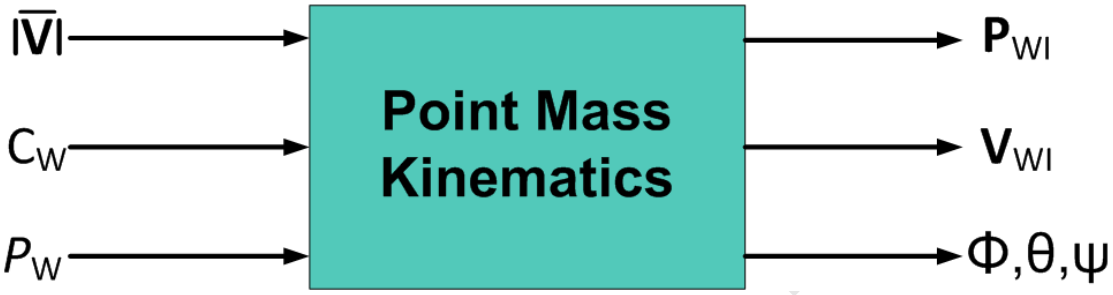

Figure 31: UAV simplified kinematics model. $\mathbf{P}_{WI}$ and $\mathbf{V}_{WI}$ are the position and velocity vectors with respect to the inertial frame.

The following assumptions have also been made for the sake of simplicity:

1) *Virtual actuators are available for the outer loop guidance algorithms* – This assumption means that the outer loops can exploit the timescale separation principle and can ignore the specific acceleration dynamics. This implies that the interface to the aircraft will be at a virtual actuator level.

2) *Airspeed is held constant* – From a guidance perspective the airspeed is not commandable but is being held constant by some arbitrary control algorithm.

Finally, the nonlinear dynamic equations are shown to be:

— — (4.10)

(4.11)

(4.12)

The reader should note that the above equations have been adapted from those derived in Gaum's thesis [39].

4.3.3 Virtual Actuator Models

According to Peddle [38], one can additionally include the dynamics of the virtual actuators when simulating the total UAV kinematics. This is of particular importance to this thesis, as the specific acceleration controllers were not explicitly designed. It is thus desirable that the behaviour of the virtual actuator controllers be modelled and the effect of their dynamics (if any) should be investigated. This will provide a more realistic simulation environment.

Peddle [38] specifies that the virtual actuators can be modelled as unity DC gain, first order lag dynamics. This translates to the effect of a low pass filter or a single pole in the s-plane. This thesis only employed the roll rate (and the NSA (virtual actuators. Thus, their transfer functions are shown as:

(4.13)

(4.14)

The theses of Gaum [39] and Blaauw [40] were investigated to determine typical bandwidths of these controllers. Additionally, Peddle [38] has provided direction on the matter.

The roll rate pole is selected to be at 25 rad/s. This implies,

(4.15)

in the s-plane. This is a reasonable assumption as Peddle states that most aircraft have a low roll Moment of Inertia (MOI) and can roll quickly [38].

The NSA pole is selected to be 10 rad/s. This implies,

(4.16)

in the s-plane. Again, Peddle shows this is a reasonable assumption as NSA dynamics are much faster than the guidance dynamics [38].

In addition to the dynamics, most virtual actuator controllers will have built in output clipping to prevent the guidance controllers from damaging the aircraft [40]. Thus, the clipping effects were also modelled.

The roll rate virtual actuator was clipped at ±180 °/s and the NSA virtual actuator was clipped at ±1.5 G. This will align with the discussions of Section 2.5 which discuss the limitations of aircraft.

The C_W virtual actuator model is shown in Figure 32. Note that the roll rate model employs the same architecture as this but for the sake of brevity will not be graphically illustrated.

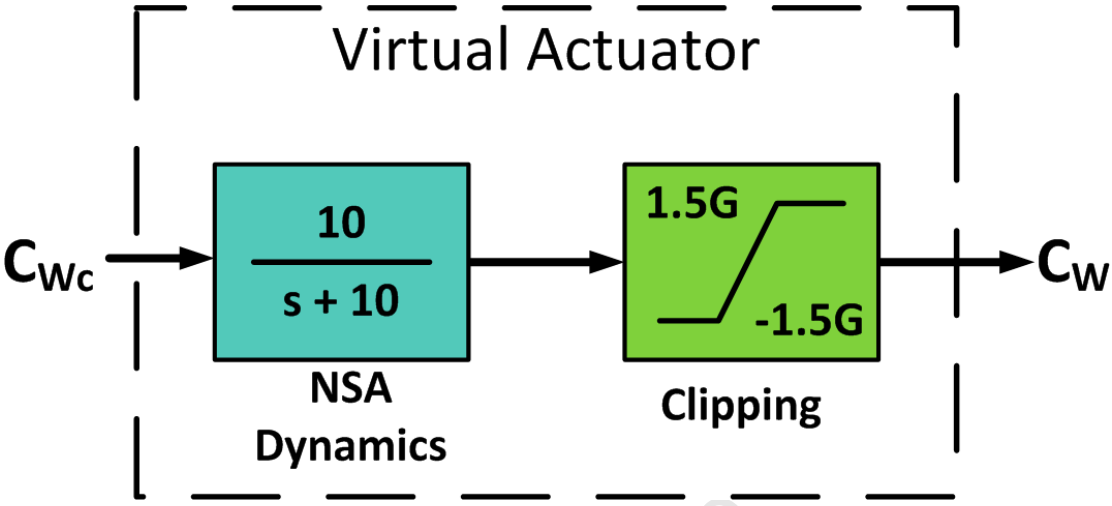

Figure 32: Virtual actuator model for NSA. The roll rate virtual actuator employs the same structure.

The virtual actuator dynamic models have been presented. The dynamics of these are much faster than the guidance dynamics and can be ignored by timescale separation when designing the guidance controllers. However, it is still desirable to observe the effects these virtual actuator dynamics will introduce when simulating the entire system.

4.4 Simulation Environment

By combining the threat kinematic model and the UAV kinematic model, a nonlinear collision encounter simulation is created. This simulation is implemented in *Simulink* and provides a test bed for the development of the CAS. This will also aid verification of the CAS in various encounter scenarios as well as quantification of its performance. *Simulink* also allows one to easily log the encounter data and rapidly plot and compare this data if required. The snapshot of the simulation developed is depicted in Figure 33.

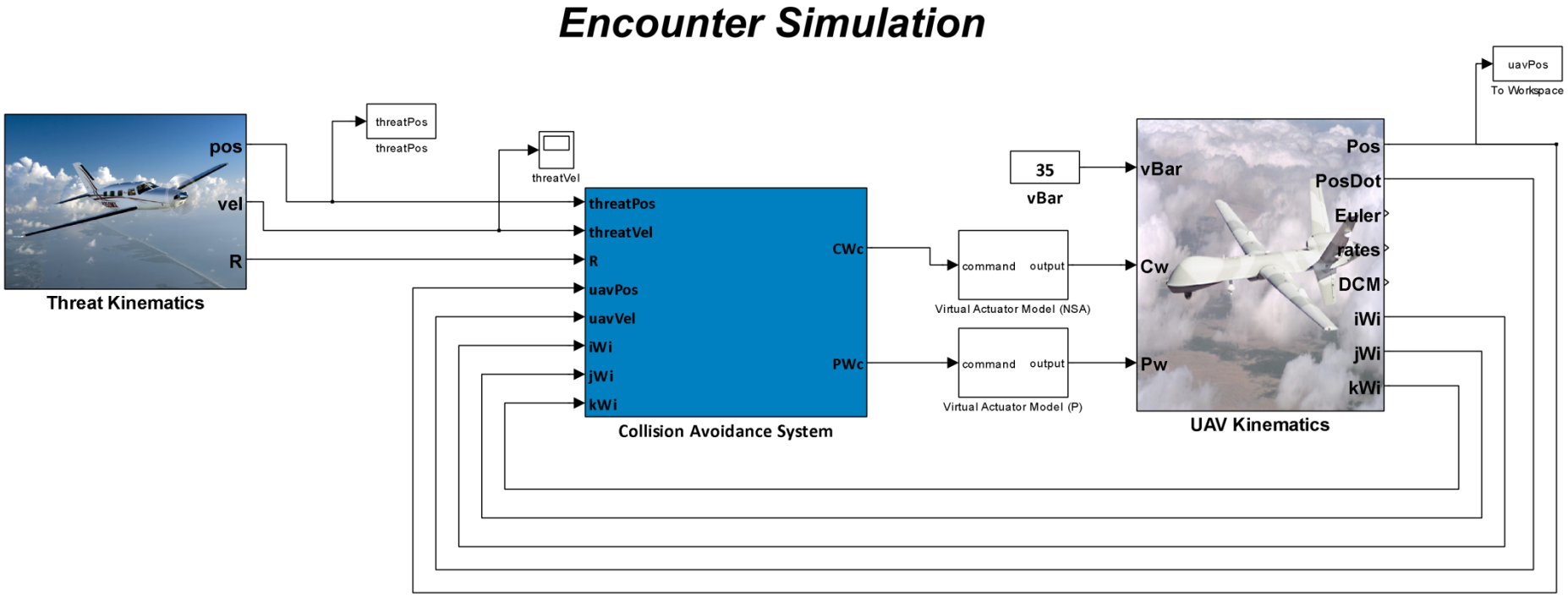

Figure 33: Encounter simulation developed in Simulink. The collision avoidance system will use the simulated model to determine evasive manoeuvres.

Chapter 5: Design and Simulation of CASSAM V.1

This chapter describes the design of the Collision Avoidance System using Specific Acceleration Matching (CASSAM). As discussed in Chapter 3, the algorithm design is in fact *iterative* and improvements will be performed. Thus, considering it is the first algorithm design, the system has been penned CASSAM V.1. In addition to the SAM Control Algorithms described previously, a new control algorithm, the *3D Velocity Guidance Controller* (3DVGC) is developed to meet the particular requirements of UAV collision avoidance.

5.1 System Overview

As stated in the previous sections, CASSAM V.1 is designed based on a combination of the SAM controller algorithms conceived by Peddle [38] and implemented in [40], [39] and [41]. In addition to these, the Collision Cone Algorithm employed by [15] is utilized to provide the reference command to the CAS.

The detailed derivations of the aforementioned algorithms will not be shown here but the reader is encouraged to consult the references if further insight is desired. The focus of this chapter will be on the implementation of these algorithms. However, the novel 3DVGC algorithm developed in this thesis will be derived and discussed in much detail.

An overview of the collection of algorithms which comprise CASSAM V.1 is shown in Figure 34. In summation, the Collision Cone Algorithm provides the reference command to the 3DVGC, which then commands a Specific Acceleration in the inertial coordinate frame. Subsequently, the Specific Acceleration Transformation Algorithm (SATA) converts this command into a corresponding NSA (C_{Wc}) and wind axis attitude command. These are then

sent to the C_W virtual actuator and Normal Specific Acceleration Vector Direction Controller (NSAVDC).

The reader should note that the green blocks represent algorithms adapted from the literature and the blue blocks represent algorithms developed in this thesis.

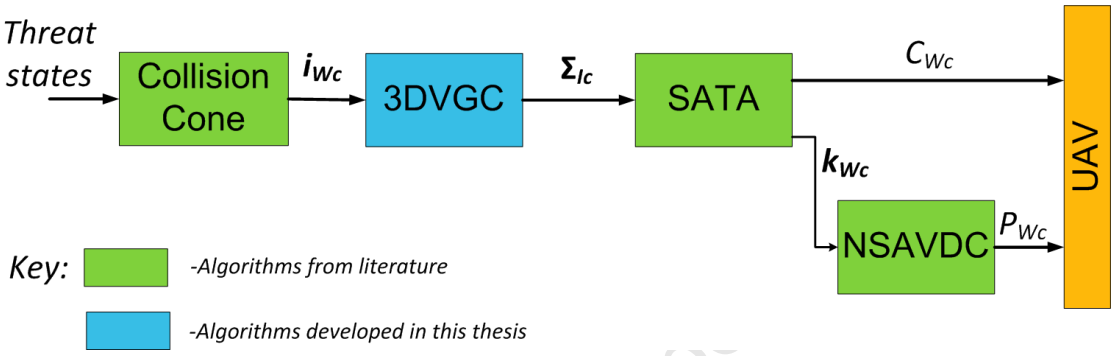

Figure 34: Overview of CASSAM V.1.

CASSAM V.1 addresses the functional requirements for sense and avoid as described in section 1.3. These are mapped back to the specific algorithms below:

- Collision Cone Algorithm addresses the Evaluate, Prioritise and Declare functional requirements.
- The other algorithms collectively address the Determine, Command and Execute functional requirements.

5.2 Collision Cone Algorithm

A Collision Cone algorithm [15], determines whether the UAV is in danger of colliding with the threat. If this is found to be true, the Collision Cone will then provide the reference command to avoid collision.

Melander [16] has shown that if we calculate the relative velocity vector as the UAV's velocity with respect to the threat, the dynamic collision avoidance problem becomes a static one. Equation (5.1) transforms the collision encounter into the encounter of a point mass with velocity and a stationary sphere.

(5.1)

Watanabe [15] has extended the Collision Cone to 3D (for spherical obstacles) by mapping the problem to a 2D plane spanned by and the relative position vector calculated by equation (5.2).

(5.2)

A special note to the reader:

It was discovered during this thesis that when and are aligned, the Collision cone algorithm breaks down. This is due to the fact the 2D plane previously is assumed by [15] to be spanned by and . Hence, when the vectors are aligned, the algorithm cannot determine this plane. This scenario was not mentioned in the literature.

A means to remedy this occurrence is to run a check before the Collision Cone algorithm is executed. This check first determines if the vectors are aligned by using the dot product to calculate the angle between the vectors. This is shown in equation (5.3).

(5.3)

If the angle equals zero, then some threshold ( ), is added to the East component of . This ensures that the 2D plane can still be calculated. In this thesis, has been intuitively selected to be 0.001.

The Collision Cone is then defined by a set of tangent lines from UAV to the "safety boundary" of the threat. The following equations are derived in [15] and [60]. However, they are repeated here for the sake of completeness. Additionally, Figure 35 will graphically aid the reader's understanding of the algorithm.

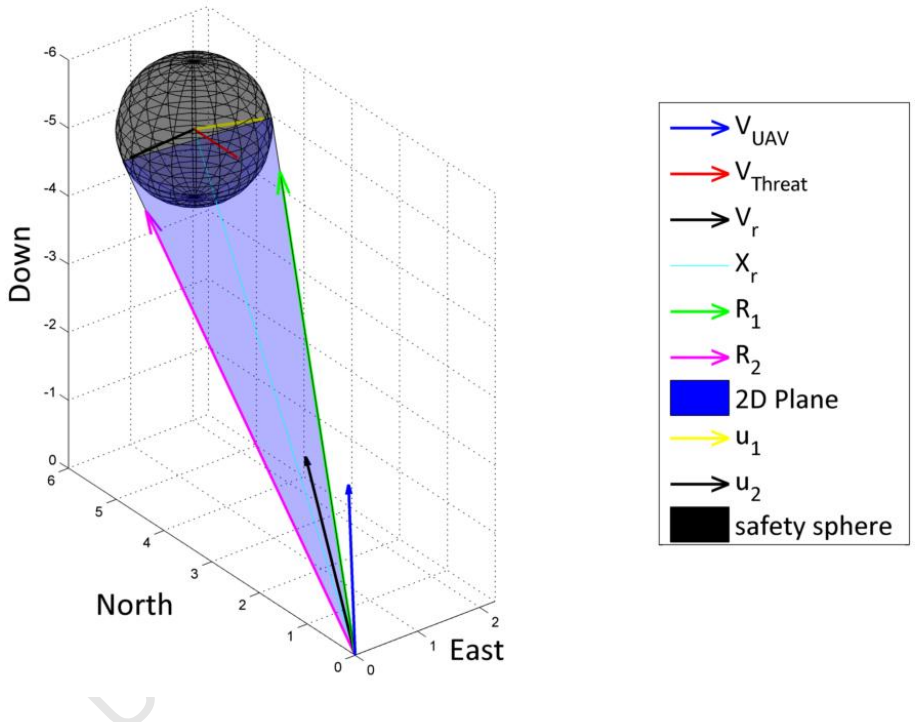

Figure 35: Collision Cone in 3D. and represent the tangent vectors bounding the cone.

Firstly, the tangent vectors and are determined by the relationship in equation (5.4). Note that .

(5.4)

Where, and are the unit vectors from the threat's position to the tangential points and is the radius of the safety sphere. These unit vectors are calculated using the following expressions:

(5.5)

(5.6)

With equation (5.7) defining .

(5.7)

Now, can be decomposed into two components: one parallel to and the other parallel to . This produces:

(5.8)

Where the coefficients of these components can be calculated as follows:

(5.9)

(5.10)

Finally, will lie in the cone if the following collision criterion [15] is true:

(5.11)

Then, to determine the closest tangent to the following relationship is provided:

(5.12)

Now that the closest tangent has been established, needs to be moved towards this vector. Watanabe [15] and Mujumdar & Padhi [10] utilised this to determine an “aiming point” for the UAV to be guided towards. However, their respective systems were only applied to stationary obstacles and a different mechanism needed to be conceived in this thesis for moving obstacles.

This is performed by first determining the rotation matrix to transform the relative velocity vector to the tangent. Considering that we have the two vectors, we can use the Axis-Angle transformation [61] explained in Appendix A.1.2. This Axis-Angle transformation can then be converted into a Rotation Matrix, . The reader should consult Appendix A.1 for methods of converting attitude representation.

Subsequently, we can apply this Rotation Matrix to the UAV’s axial unit vector ( ). This will result in the direction which the UAV’s total velocity vector should be steered to move out of the Collision Cone. Mathematically, this shown by equation ((5.13) and graphically illustrated by:

(5.13)

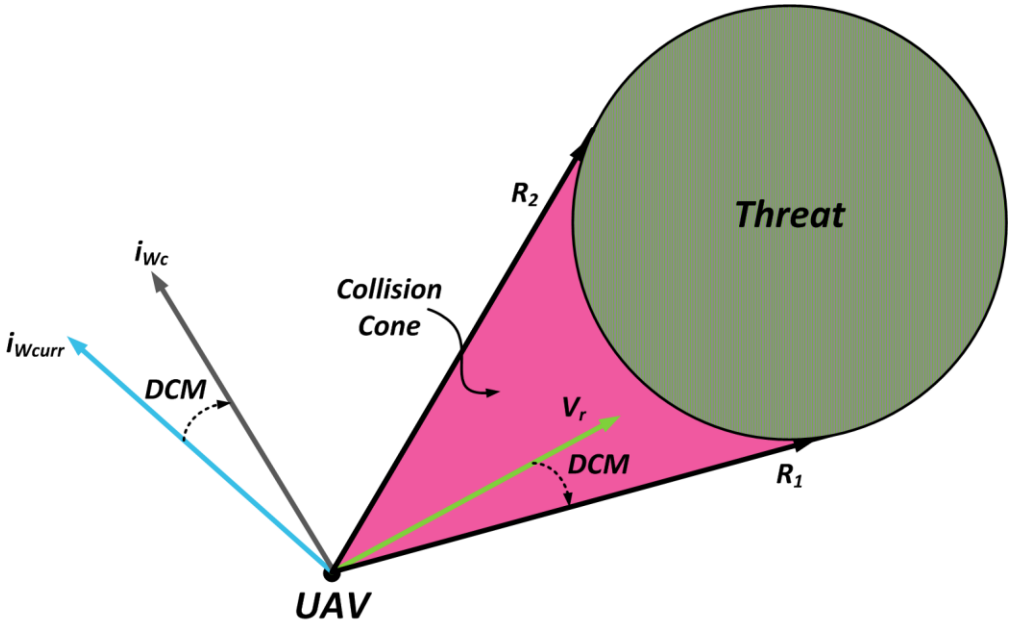

Figure 36: A Rotation Matrix (DCM) is determined from $\mathbf{V_R}$ to $\mathbf{R_1}$. This DCM is then used to rotate $\boldsymbol{i_{Wcurr}}$ to the new velocity direction, $\boldsymbol{i_{Wc}}$.

This new total velocity direction, , is the direction that the UAV must steer to avoid collision.

An additional note is that this algorithm is executed at 1 kHz. This will ensure that the velocities will remain approximately constant within the sample time. This aligns with the constant velocity assumption of the original Collision Cone proposed by Chakravarthy and Ghose [11].

The Collision Cone algorithm implemented in CASSAM V.1 has been described. It is based primarily on the work done by [15] and [10] but has been augmented to function with moving obstacles. The algorithm outputs a commanded total velocity direction which will be employed by the 3DVGC to avoid collisions.

Although not explicitly mentioned, the Collision Cone Algorithm addresses the functional requirements described in Section 1.3. The Evaluate and Declare functional requirements are satisfied by the fact the Collision Cone algorithm

evaluates the risk of collision and if a collision is imminent, it declares this by sending a reference command to the 3DVGC.

The Prioritise functionality however, was not directly implemented. But, it can be argued that the encounter scenarios only contained one threat. Thus, there was no need to prioritise threats in this thesis. The reader should note that though the Prioritise functionality was not implemented, it has been *designed for*. This can be achieved in the future application to address multiple threats, by using the angular width of the cone. With this information one can determine which threat poses the greatest risk and *prioritise* accordingly.

The forthcoming subsection will describe the SAM Controllers which form part of CASSAM V.1.

5.3 SAM Controllers

The SAM Architecture greatly simplifies the guidance problem for UAVs [38]. The reason for this is that it allows the aircraft model to be reduced to a steerable acceleration vector. This concept was greatly discussed in Section 2.4 and it is this simplification which will be exploited by CASSAM V.1. The detailed design and derivation of these algorithms has already been provided Peddle [38] and the reader has been given conceptual insight in Chapter 2. Accordingly, the focus of this subsection will be on how these algorithms were implemented in CASSAM V.1.

5.3.1 Normal Specific Acceleration Vector Direction Controller

The Normal Specific Acceleration Vector Direction (NSAVDC) controller rotates the NSA to the required orientation. It performs this by calculating the error angle between the commanded and the current and then drives this to zero by commanding a Wind-Axis roll rate ( ).

Peddle [38] states that one can design this controller in two ways. It can be designed by factoring in the roll rate virtual actuator dynamics or by assuming a timescale separation between the NSAVDC and the roll rate controller dynamics. The latter method is chosen for CASSAM V.1 as it is simpler and as stated by Peddle, the roll moment of inertia of most aircraft is very small, making this a valid assumption.

The control law is illustrated in equation (5.14).

(5.14)

Because the dynamics of the error angle consists of a single integrator on the origin, the error angle pole can be placed by selecting an appropriate feedback gain k_{φ}.

Considering Figure 37, the error angle pole is constrained in the s-plane. The location of this pole is limited from above (green region) by assuming the roll rate virtual actuators timescale separation. It is also limited from below (purple region) by the timescale separation to guidance controller dynamics.

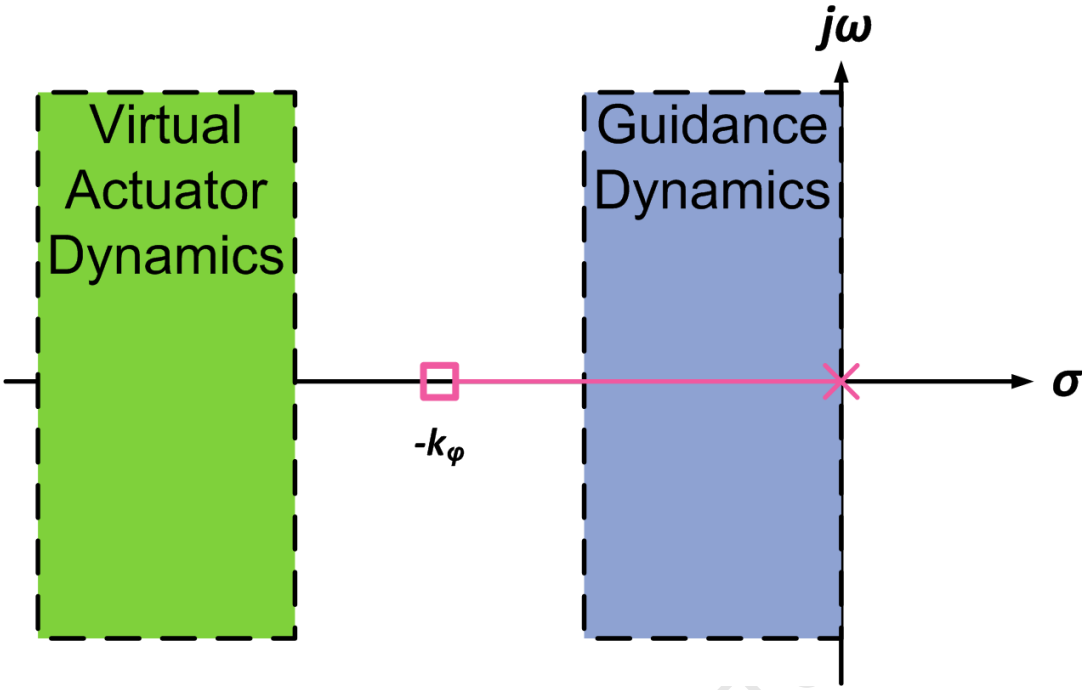

Figure 37: Conceptual root locus diagram for the NSAVDC. This figure illustrates the factors affecting the selection of the error angle pole.

The reader is reminded, in section 4.3.3 the roll rate virtual actuator dynamics were chosen to be 25 rad/s. Considering that Peddle specifies a timescale separation factor of five is sufficient [38], the error angle pole is selected to be 5 rad/s. Thus,

(5.15)

will ensure that enough time separation exists from the virtual actuator. This will also allow the fastest guidance dynamics to have a bandwidth of 1 rad/s.

The error angle is calculated using (5.16). Please note that φ is not necessarily equivalent to the bank angle [38].

(5.16)

The NSAVDC has now been designed. This controller will orientate the NSA vector by commanding a roll rate to drive the error angle to zero. This algorithm is fed a command produced by the Specific Acceleration Transformation Algorithm.

5.3.2 Specific Acceleration Transformation Algorithm

The Specific Acceleration Transformation Algorithm (SATA), matches the specific acceleration commanded in the inertial frame ( ) to an equivalent Specific Acceleration in the wind axis frame given the constraint of zero sideslip.

The overall functionality of this algorithm has been explained in Chapter 2. However, in this subsection the reader will be given insight into the implementation of the SATA in CASSAM V.1.

Peddle [38] has shown that the commanded specific acceleration is defined to lie within the aircraft X_W-Z_W plane and can be written as,

(5.17)

with and are the commanded axial and normal specific accelerations by the guidance controller. Also, is the commanded NSA vector direction.

However, as was demonstrated by Gaum [39], the bandwidth limited nature of the ASA virtual actuator renders it unfeasible for guidance control as it does not hold the timescale separation. Consequently, Gaum has shown that this component can be removed from the commanded specific acceleration vector. The reader will note that this has been conceptually demonstrated in Figure 18 previously in Chapter 2.

From Gaum's thesis [39], the ASA component is "extracted" by using the dot product in equation (5.18).

(5.18)

This component can be subtracted from the commanded specific acceleration to produce the commanded NSA Vector. This is shown in equation (5.19).

(5.19)

Now, that the commanded NSA vector has been calculated. The corresponding commands to the NSA controller (virtual actuator) and NSAVDC can be extracted by using the following,

(5.20)

(5.21)

where is defined in [38] as a reference vector which can allow for inverted flight by swapping the sign of the commanded NSA. For CASSAM V.1 we will *not* be employing inverted flight and thus, is selected to be the current unit vector of the Z_W component ( ).

The design of the SATA for CASSAM V.1 has been presented. In summation, the algorithm matches the specific acceleration commanded in the inertial frame ( ) to an equivalent specific acceleration in the wind axis frame. It does this by commanding an appropriate NSA vector ( ) and a direction vector . Essentially, to obtain a specific acceleration the UAV will roll and

command an applicable NSA. This is shown in graphically in a figure by Gaum [39] repeated here as Figure 38.

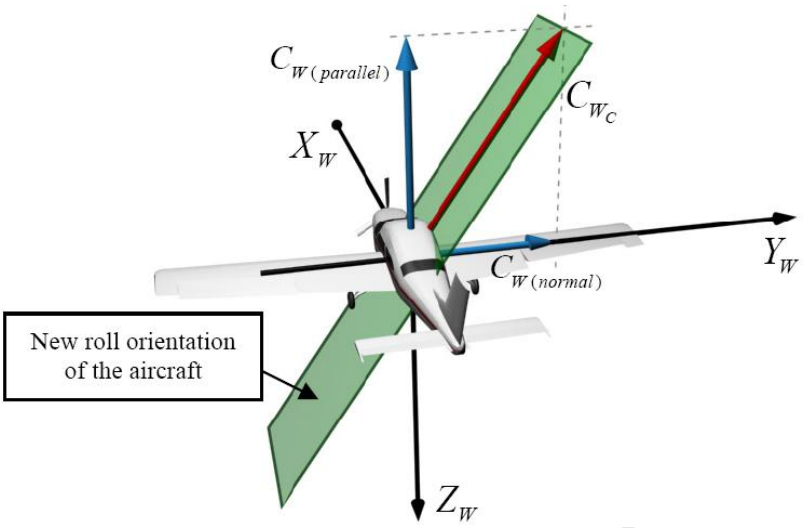

Figure 38: Given a commanded specific acceleration in inertial, the SATA will command the aircraft to roll and command an applicable C_{Wc} [39].

The SAM Architecture is normally employed with outer loop commands being generated by an inertial Position or Velocity control loop [38], [39], [41]. Unfortunately, these loops command a change in total velocity, which will result in a requirement for a corresponding change in airspeed and in turn, an axial acceleration. Suitably, due to the bandwidth limits on the ASA, CASSAM V.1 was expected not to employ this virtual actuator in its design.

As a result of this constraint, a different approach needed to be considered for CASSAM V.1; the 3D Velocity Guidance Controller addresses this issue.

5.4 3D Velocity Guidance Controller

As discussed in the foregoing section, for CASSAM V.1, a change in airspeed must be circumvented as it will require an axial acceleration, which has been proven to be bandwidth limited. As a consequence, slowing down or speeding up the UAV will not be an option to avoid collisions

The dilemma is then how to guide the relative velocity vector without changing the magnitude of the UAV's total airspeed vector. This is precisely what the 3D Velocity Guidance Controller (3DVGC) does. It "steers" the total velocity vector by commanding a total acceleration normal to the airspeed vector. Hence, the aircraft's axial unit vector ($\boldsymbol{i_W}$) can effectively be steered towards the Collision Cone's edge, thereby avoiding the collision.

The reader should note that this algorithm (in combination with the SAM Controllers) addresses the functional requirements of Determine, Command and Execute as described in section 1.3.

The 3DVGC is comprised of two algorithms: The Direction Vector Algorithm (DVA) and a control algorithm. This is shown in Figure 39. The DVA is responsible for determining the direction vector for the total acceleration command. The control algorithm is responsible for calculating the error angle and then driving this error to zero.

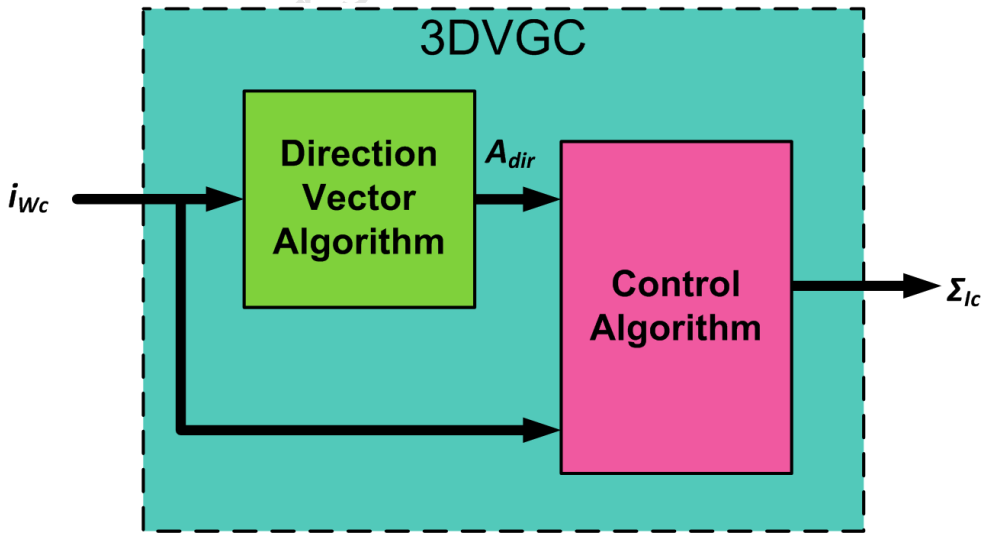

Figure 39: the 3DVGC is comprised of two subfunctions: The DVA and a control algorithm.

The 3DVGC is engaged when enabled by the Collision Cone algorithm described previously. When not engaged, CASSAM V.1 simply sends a specific acceleration command of [0 0 -9.81] m/s^2 to counteract the effect of gravity to SATA. This will keep the aircraft flying straight and level.

In the following subsections, the essence of the 3D Velocity Guidance Controller is described.

5.4.1 Direction Vector Algorithm

As mentioned previously the DVA is responsible for determine the direction vector in which the total acceleration will act. Firstly, we utilize the current $\boldsymbol{i_W}$ unit vector of the Wind-Axis ( ) and cross product it with the desired $\boldsymbol{i_W}$ unit vector ( ) – This will give the vector perpendicular to the plane (**a**) in which the required total acceleration vector (**A**) will lie. This is mathematically given by equation (5.22).

(5.22)

To obtain the direction vector ($\boldsymbol{A_{dir}}$) along which the total acceleration vector will lie, the cross product of and **is taken**.

(5.23)

This result gives us the unit vector along which the total acceleration vector will lie. Because ***A*** will always lie perpendicular to , it will not have an effect on the airspeed magnitude.

The unit vectors employed by the DVA are shown below:

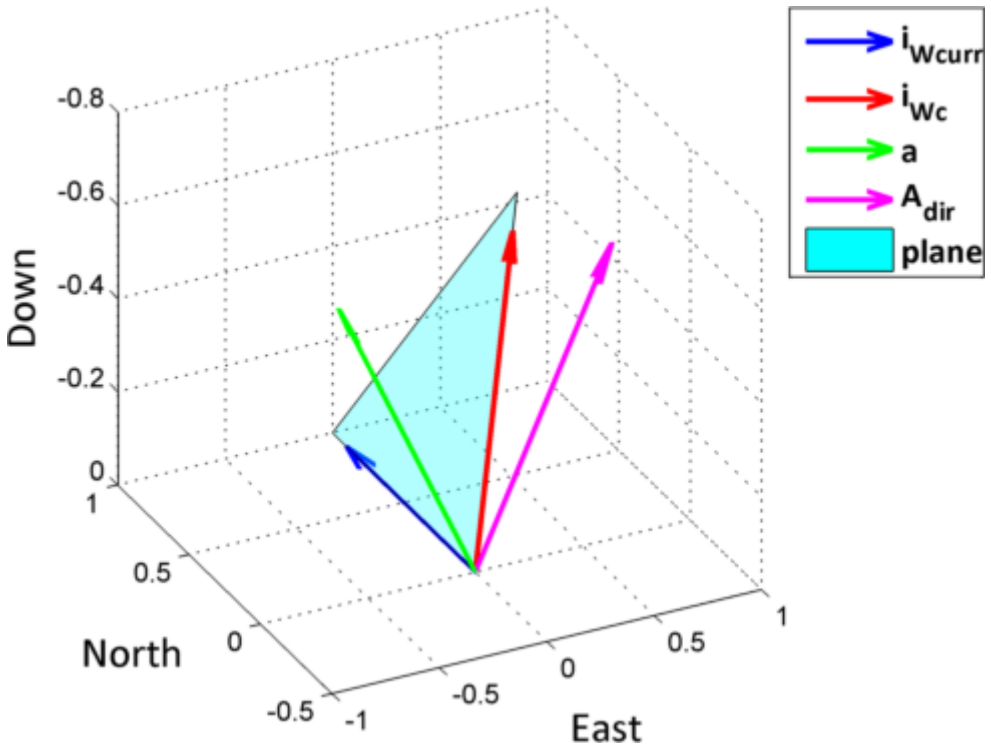

Figure 40: Unit vectors employed by the DVA to determine the direction of the total acceleration vector.

5.4.2 Control Algorithm

The control algorithm is responsible for calculating the error angle between the command axial unit vector and the current axial unit vector. Using this angle, it will command an appropriate total acceleration.

The magnitude of total acceleration ***A*** is determined by the size of the error angle ( ). This angle is defined in as:

(5.24)

It is then the task of the control algorithm to drive this angle to zero by commanding an applicable magnitude of ***A***.

For simplicity, the error angle dynamics are assumed to be linear and defined by the following relationship:

– (5.25)

The total acceleration magnitude (***A***) is also assumed to be approximately equal to the derivative of the error angle. This is shown below in equation (5.26).

(5.26)

This assumption is justified by the fact that the total acceleration vector ***A*** will always be acting in the plane containing . With reference to Figure 41, it can be argued that if the total acceleration is commanded along $\boldsymbol{A_{dir}}$, the current $\boldsymbol{i_{Wcurr}}$ will rotate towards the commanded . This will cause the error angle to decrease proportionally. This justifies the relationship defined by equation (5.26) to be a reasonable assumption.

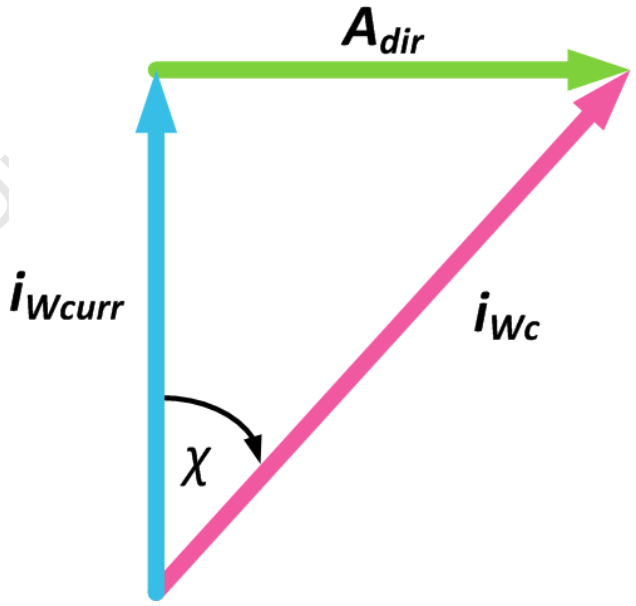

Figure 41: Unit vector diagram used to justify of error angle dynamics. We can see that the $\boldsymbol{A_{Dir}}$ unit vector acts in a direction which will reduce the error angle.

In the s-plane, this implies a single integrator on the origin. The 3DVGC dynamics can then be set to the desired closed loop positions. However, these are limited from above by the bandwidth of the NSAVDC and the NSA virtual actuator. Assuming the NSAVDC to be slower than the NSA dynamics, the fastest guidance dynamics as described in section 5.3.1 will be 1 rad/s.

The control topology selected to drive to zero is the familiar PI control loop. The reason for this is that the can be approximate as a ramp reference command. This is due to the Collision Cone widening at each time instant during the approach of the threat as shown in Figure 42.

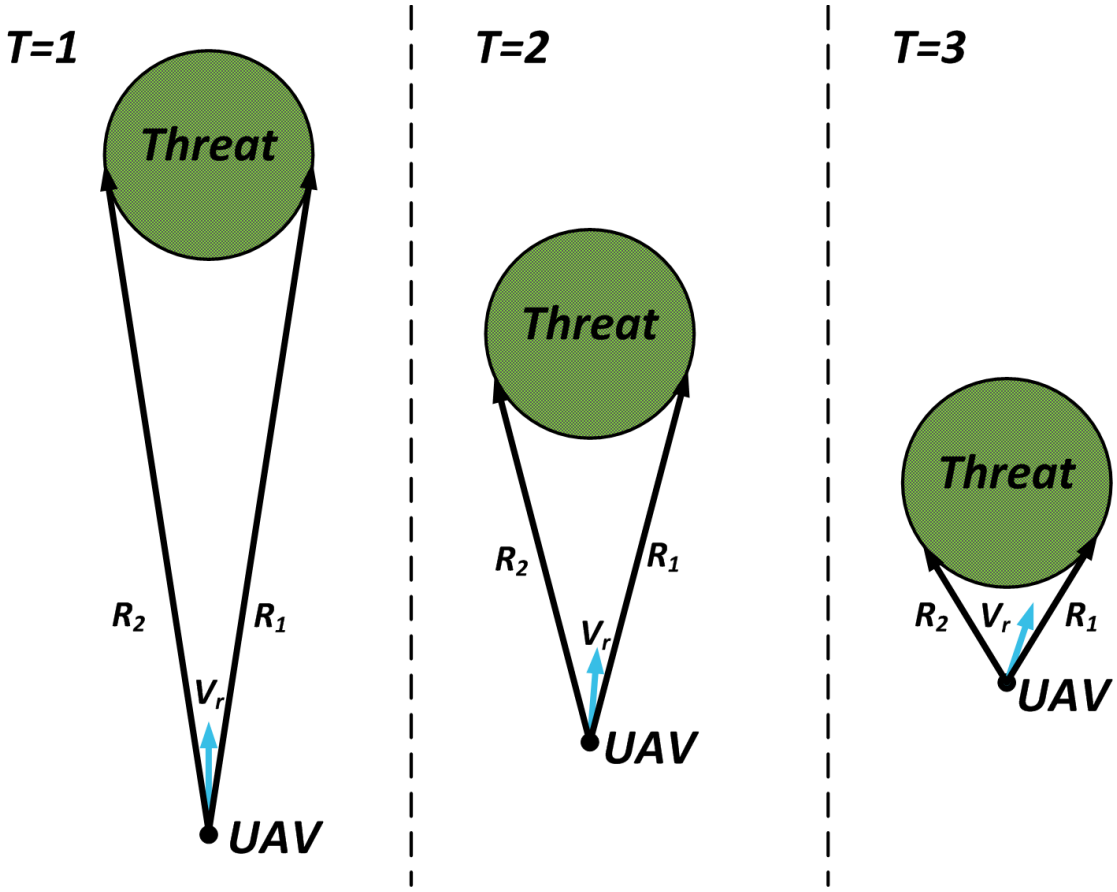

Figure 42: The Collision Cone widens as the threat moves closer to the UAV. Thus, the reference command to the 3DVGC is assumed to be a ramp.

Thus, if the controller needs to track this reference command with zero error it requires two free integrators. From equation (5.25) it is evident that there exists only one free integrator in the dynamics. Therefore, the correct system type number is achieved by adding the additional integrator of the PI Control Loop. The control algorithm is architecture is shown below:

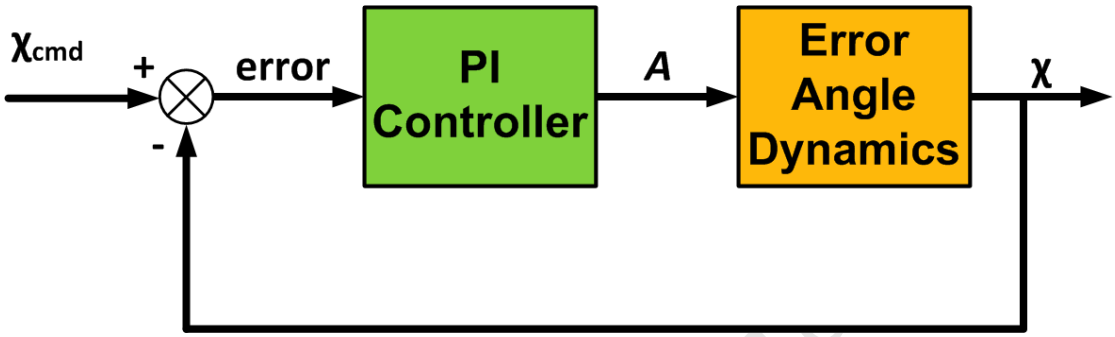

Figure 43: Control architecture employed by the 3DVGC. The PI Controller uses the error to command an applicable total acceleration (A).

PI Controllers [62] take the general form of,

— (5.27)

Where, is the proportional gain and is the integral gain. This, when combined with the open loop dynamics described by (5.25) results in the following closed loop characteristic equation [62],

(5.28)

It is evident that the system consists of two poles and because a PI topology provides us two degrees of freedom, the poles can be placed on any desired location in the s-plane.

The desired closed loop poles are selected to be at . These will provide a sufficient damping factor of 0.7071. The dominant frequency of this closed loop system is also 5 times slower than that of the NSAVDC pole designed in subsection 5.3.1, which is located at . Thus, the timescale separation can be assumed to hold true.

The desired characteristic equation is then:

(5.29)

Finally, by equating equation (5.28) and equation (5.29), the resulting gains are:

(5.30)

(5.31)

The root locus is plotted in Matlab and illustrated by Figure 44.

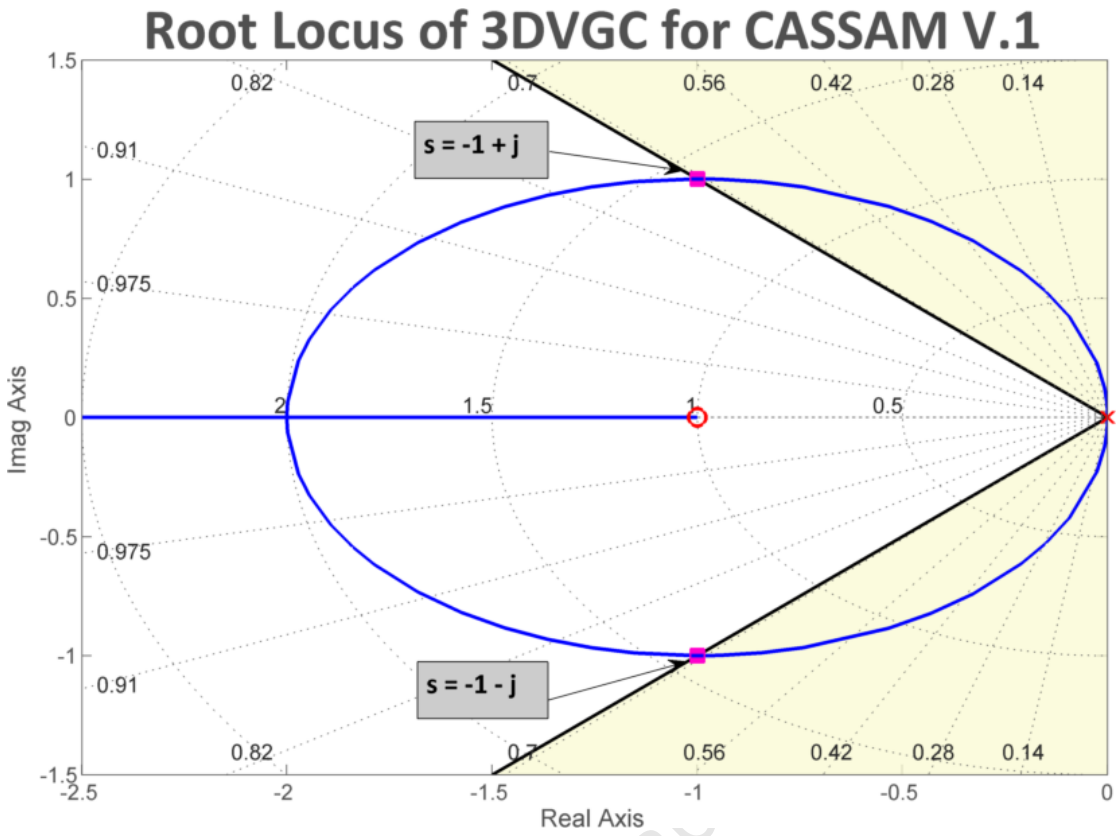

Figure 44: Root Locus for the 3DVGC. The diagram illustrates the closed loop poles have been moved to the desired positions (the pink squares). Also, evident is that the poles are adequately damped.

The 3DVGC has now been designed. The controller employs a PI control algorithm to drive the error angle to zero. Pole placement has been employed to move the closed loop poles to the desired locations.

The 3DVGC was now integrated with the Collision Cone and SAM Controller algorithms to form CASSAM V.1. These were implemented in Simulink. The following subsection will reveal the simulation test results for CASSAM V.1.

5.5 Simulation Results

The aforementioned algorithms were then implemented in Simulink and integrated into the main simulation described in Chapter 4. CASSAM V.1 was then tested under the discrete scenarios described in the Methodology and the results are depicted below.

Note in the forthcoming figures, the Closest Point of Approach (CPA) is annotated in the diagram by the square and circle. To achieve a means of illustrating the motion between the aircraft, T(CPA)-1 and T(CPA)+1 are depicted as well. These are the time instants one second before and after the CPA respectively.

5.5.1 Head on Collision (S0) - TTI 20s

The initial states are depicted in Table 2. The reader should note that all vectors are in the NED frame.

Table 2: Initial States for S0 with TTI 20s for CASSAM V.1.

Initial States	Velocity (m/s)	Position (m)	Heading (°)
UAV	[35 0 0]	[0 0 0]	0
Threat	[-100 0 2]	[2700 0 – 40]	180

The resulting trajectory plot is illustrated by Figure 45. Note that the blue dot and red sphere depict the positions of the UAV and threat at t(CPA).

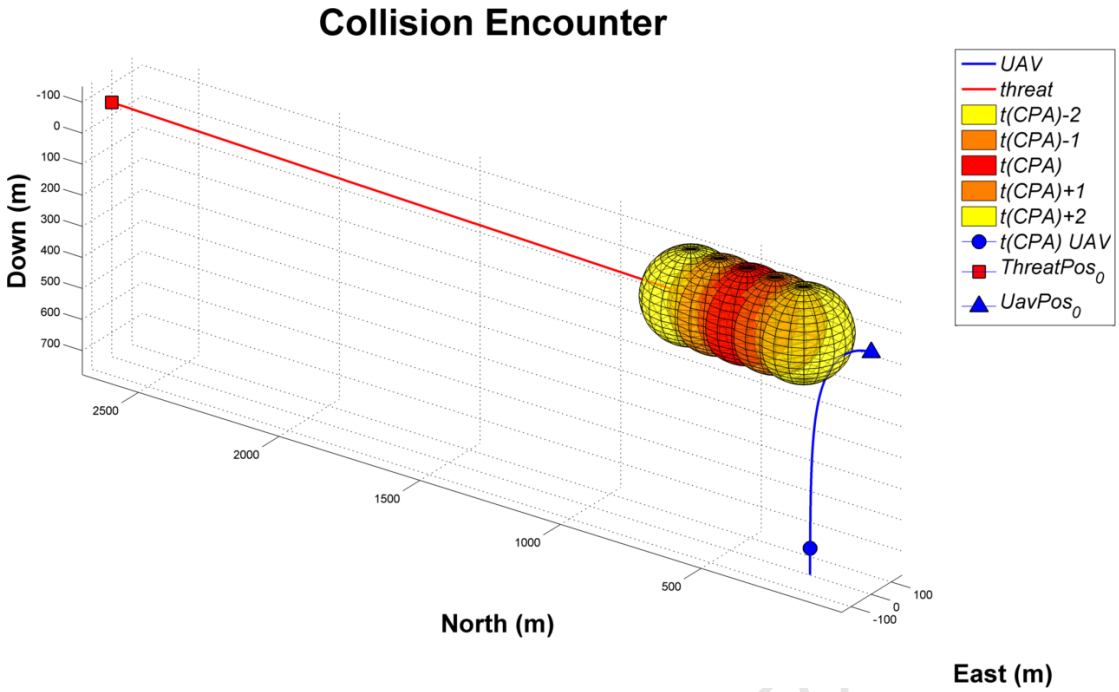

Figure 45: Trajectory plot for S0 with TTI 20s for CASSAM V.1. It is evident that the threat has been avoided.

Also depicted graphically in Figure 46 are the resulting attitude angles, and commanded NSA. The reader will also note the Collision Cone trigger is also shown. This trigger provided an indication of the time taken to guide the UAV to safety.

From the data, it is evident that the UAV dove straight down to a pitch angle of 90° to successfully avoid collision. It also utilized approximately zero bank angle to achieve this. The commanded NSA was close to zero as the UAV had entered free fall. The TTI was 0.88s.

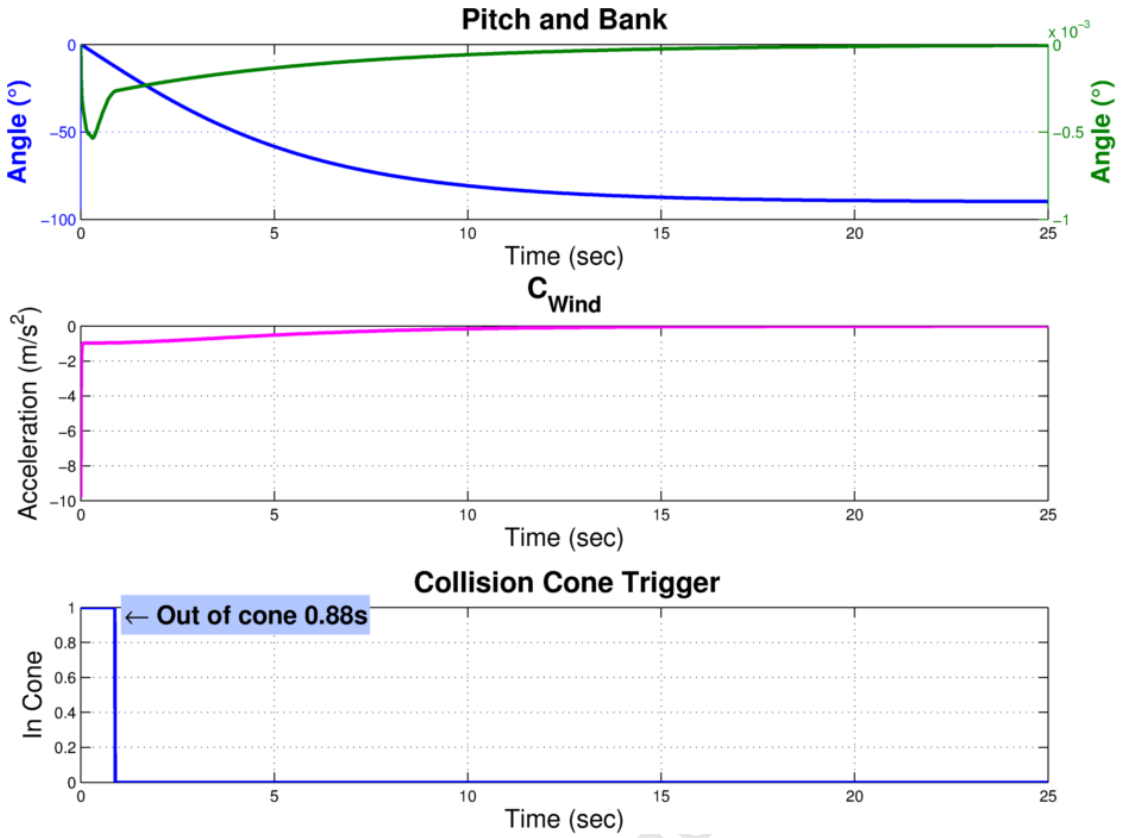

Figure 46: Data captured for S0 with TTI 20s. The Time-to-Safety was 0.88s.

5.5.2 Side on Collision (S1) - TTI 20s

The initial states are depicted in Table 1. The reader should note that all vectors are in the NED frame.

Table 3: Initial States for S1 TTI 20s for CASSAM V.1.

Initial States	Velocity (m/s)	Position (m)	Heading (°)
UAV	[35 0 0]	[0 0 0]	0
Threat	[0 -120 -2]	[700 2400 40]	-90

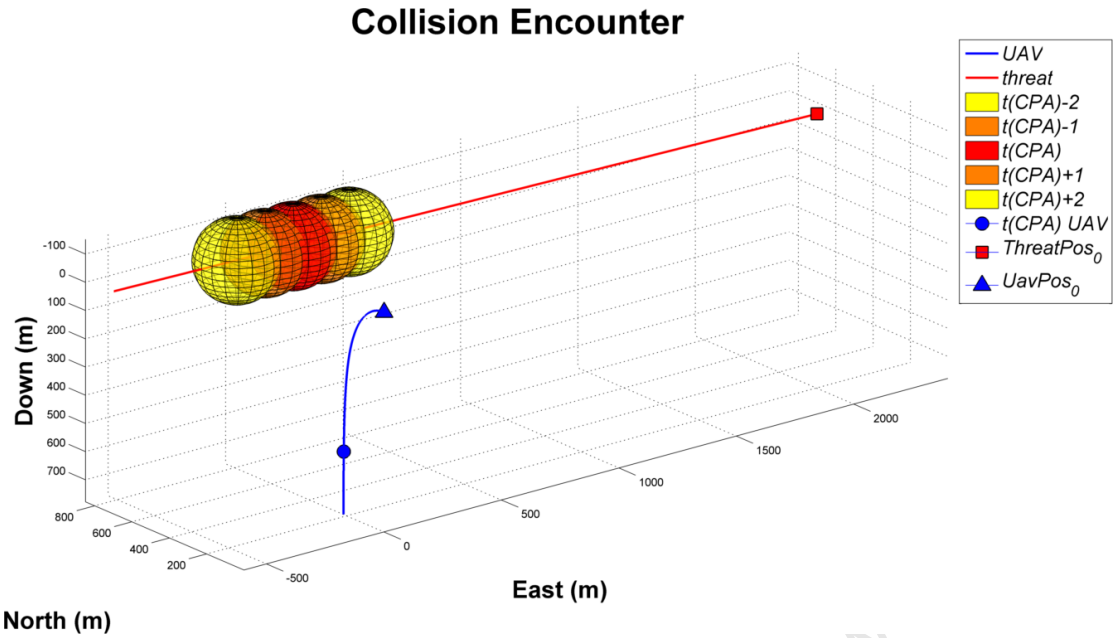

Figure 47: Trajectory plot for S1 with TTI 20s for CASSAM V.1. It is evident that the threat has been avoided.

The resulting trajectory plot is illustrated by Figure 47. Note that the blue dot and red sphere depict the positions of the UAV and threat at t(CPA).

The data captured is illustrated in Figure 48. It is shown that once more UAV dove straight down to a pitch angle of 90° to successfully avoid collision. It also utilized negligible bank angle in this evasive manoeuvre. The commanded NSA was close to zero as the UAV had entered free fall. The TTI was 0.88s.

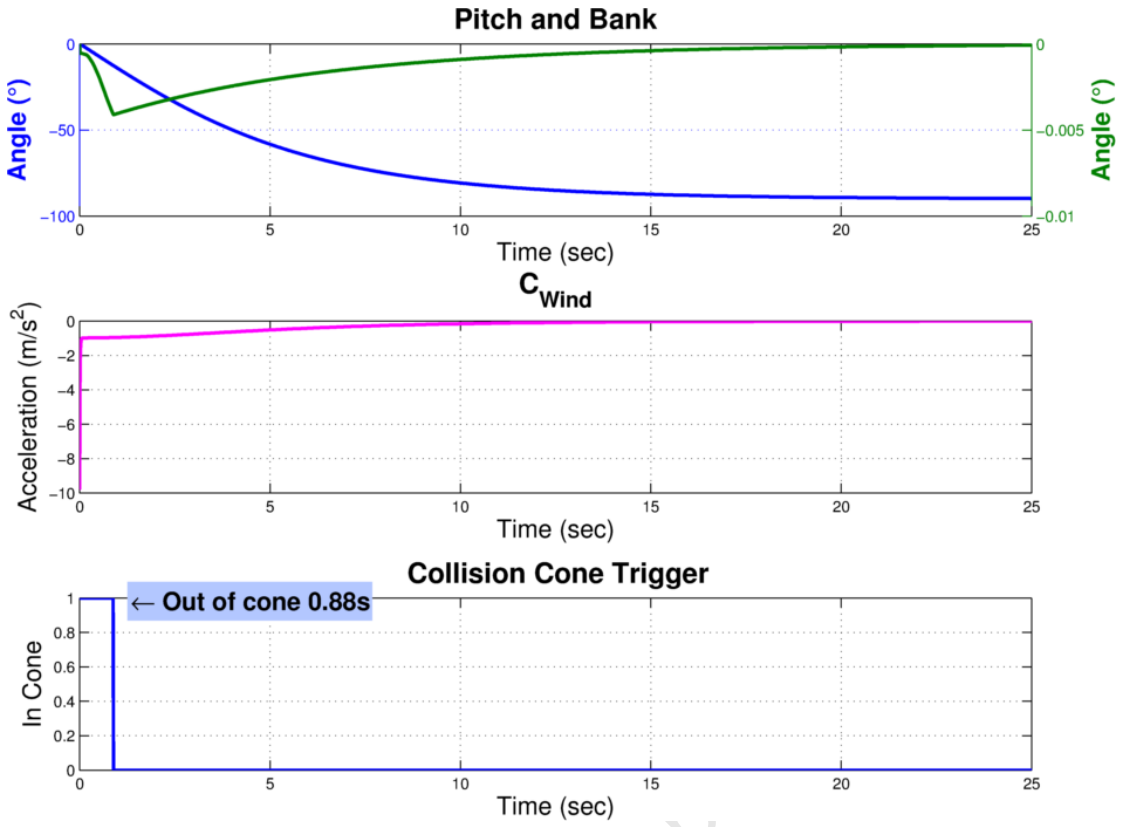

Figure 48: Data captured for S1 with TTI 20s. The Time-to-Safety was 0.88s.

5.5.3 Head on Collision (S0) - TTI 10s

The initial states are depicted in Table 4. The reader should note that all vectors are in the NED frame.

Table 4: Initial States for S0 with TTI 10s for CASSAM V.1.

Initial States	Velocity (m/s)	Position (m)	Heading (°)
UAV	[35 0 0]	[0 0 0]	0
Threat	[-100 0 2]	[1350 0 -20]	180

The resulting trajectory plot is illustrated by Figure 49. Note that the blue dot and red sphere depict the positions of the UAV and threat at t(CPA).

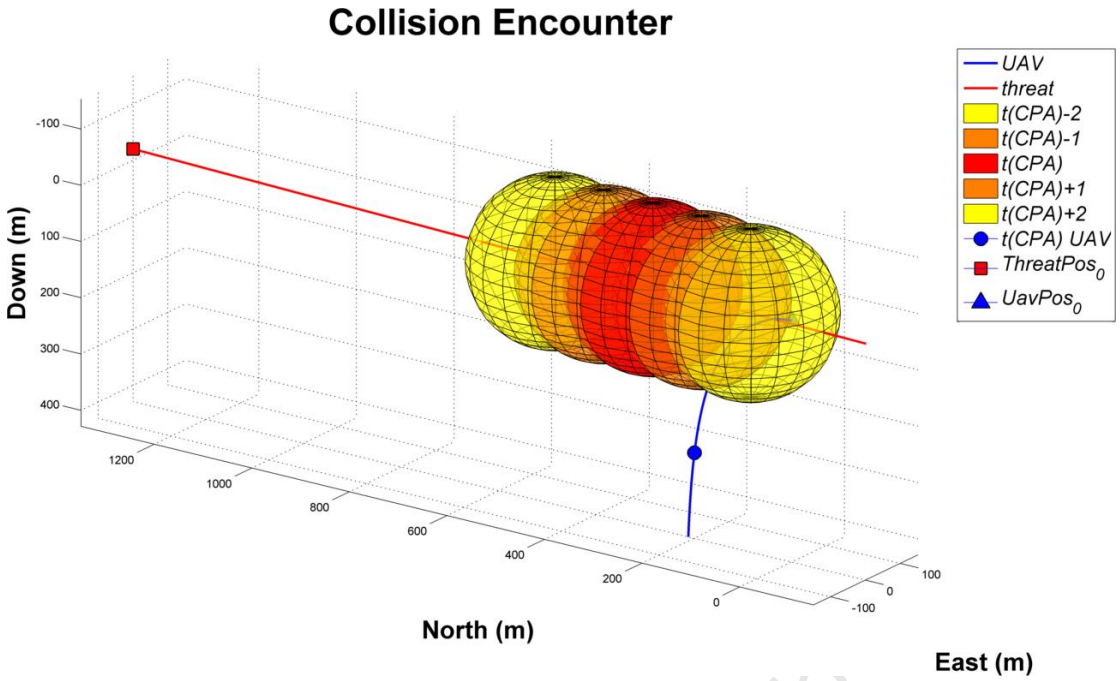

Figure 49: Trajectory plot for S0 with TTI 10s for CASSAM V.1. It is evident that the threat has been avoided.

The data captured for this scenario is shown in Figure 50. Much like the previous encounter scenarios, the UAV pitched down immediately to avoid the collision. Similarly, minimal bank angle and NSA were commanded. One difference to the previous simulations is the TTI increased to 2s.

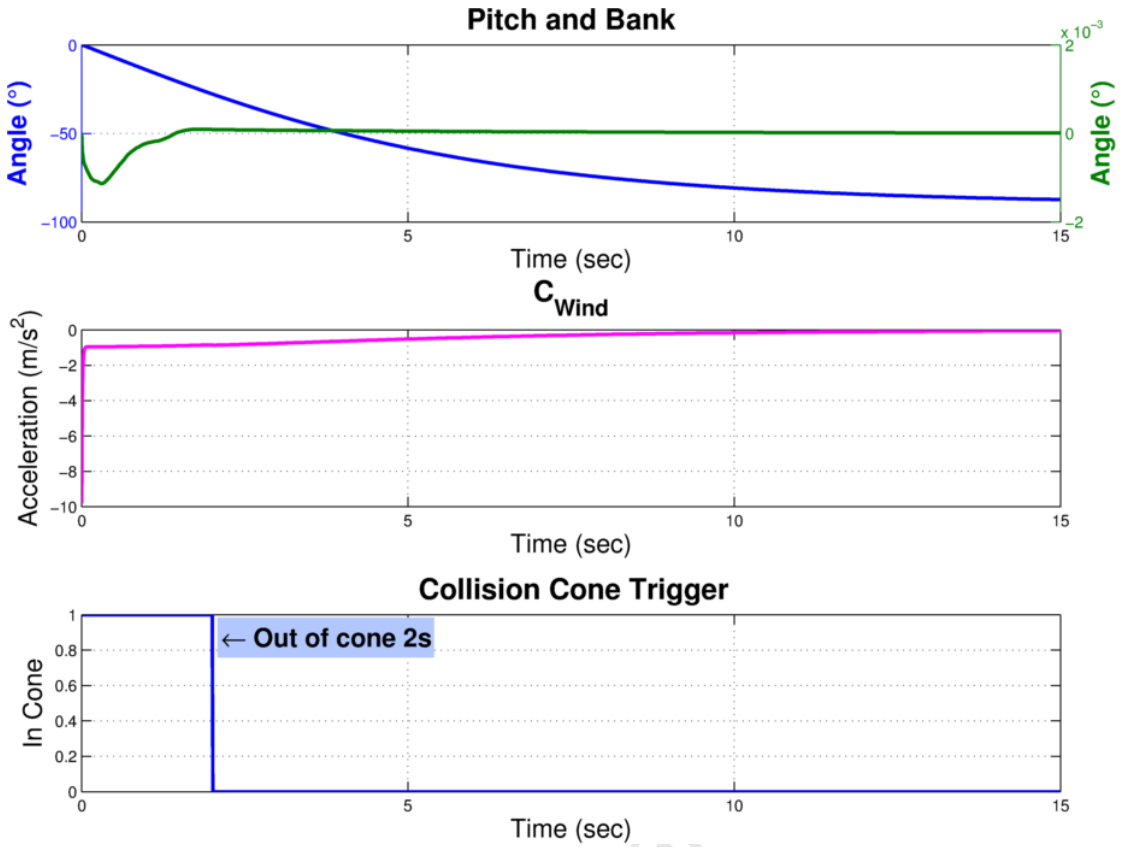

Figure 50: Data captured for S0 with TTI 10s. The Time-to-Safety was 2s.

5.5.4 Side on Collision (S1) - TTI 10s

The initial states are depicted in Table 5. The reader should note that all vectors are in the NED frame.

Table 5: Initial States for S1 TTI 10s for CASSAM V.1.

Initial States	Velocity (m/s)	Position (m)	Heading (°)
UAV	[35 0 0]	[0 0 0]	0
Threat	[0 -120 -2]	[350 1200 20]	-90

The trajectory plot is shown by Figure 51. Once more, the UAV successfully avoided the threat.

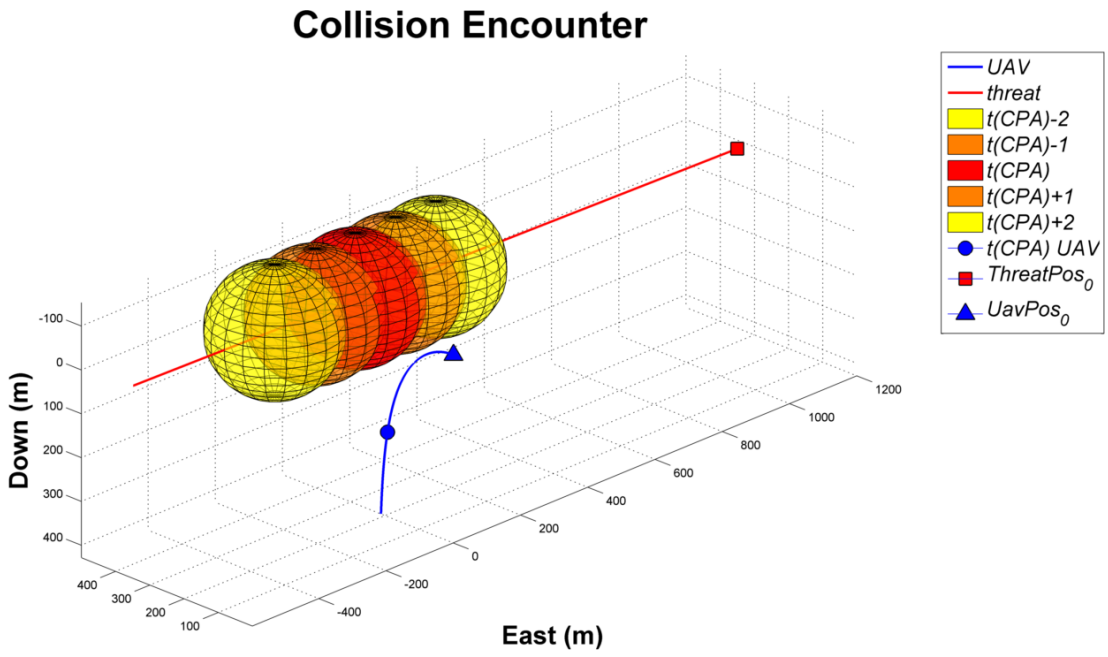

Figure 51: Trajectory plot for S1 with TTI 10s for CASSAM V.1. It is evident that the threat has been avoided.

The data captured is illustrated in Figure 52. In a similar manner to the previous simulations, the UAV pitched down to 90° to evade collision. Also, negligible bank angle and NSA were commanded. The TTI increased from the previous S1 encounter to 1.99s.

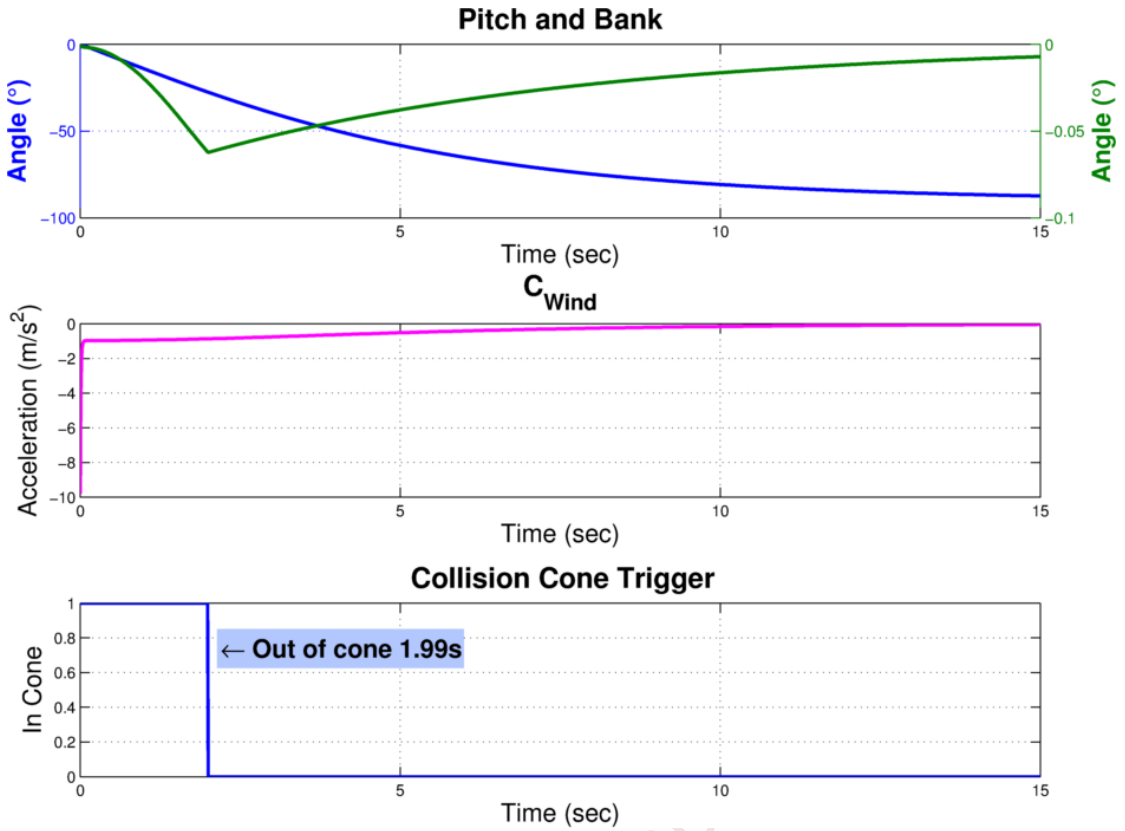

Figure 52: Data captured for S1 with TTI 10s. The Time-to-Safety was 1.99s.

5.5.5 Summary

In summary, CASSAM V.1 evaded the threats in all of the test scenarios. In all of the scenarios it pitched straight down 90° into a dive. This was the direction determined by the Collision Cone Algorithm as the best manoeuvre.

It also utilized a negligible bank angle during these evasive manoeuvres. The NSA commanded was also always zero to pitch the aircraft down rapidly. The time to safety (TTS) also increased with a decrease in TTI.

Chapter 6: Design and Simulation of CASSAM V.2

It was noticed in the previous chapter that CASSAM V.1 did not behave as expected. Furthermore, the system often commanded the UAV to dive rapidly to avoid collision. However, this manoeuvre strategy may not be the desirable for the aircraft as discussed in Section 2.5. This will be discussed in more detail in Chapter 8. Furthermore, the error angle dynamics for the 3DVGC were investigated more thoroughly as the algorithm did not behave as effectively as was expected.

Therefore, several augmentations to the original CASSAM V.1 needed to be made to conceive a possibly improved system. The adaptations to CASSAM V.1 are described in what follows and the new system is known as CASSAM V.2.

6.1 System Overview

Conceptually, CASSAM V.2 operates in the same manner to CASSAM V.1. In fact, the architecture depicted in Figure 34 still applies. The only differences are in the algorithms themselves. The principal adaptations are described below:

- The Collision Cone Algorithm was augmented to produce the Projected Collision Cone Algorithm (PCCA) as a means to prevent the UAV from only diving or climbing to avoid collision.
- The 3DVGC error dynamics were thoroughly investigated and a new algorithm, the Nonlinear 3DVGC, was implemented.

In addition to these, minor adaptations were also made to the NSAVDC to enable it to clip the commanded bank angle. The SATA was left unaltered.

For the sake of completeness, a diagram of the architecture is illustrated in Figure 53.

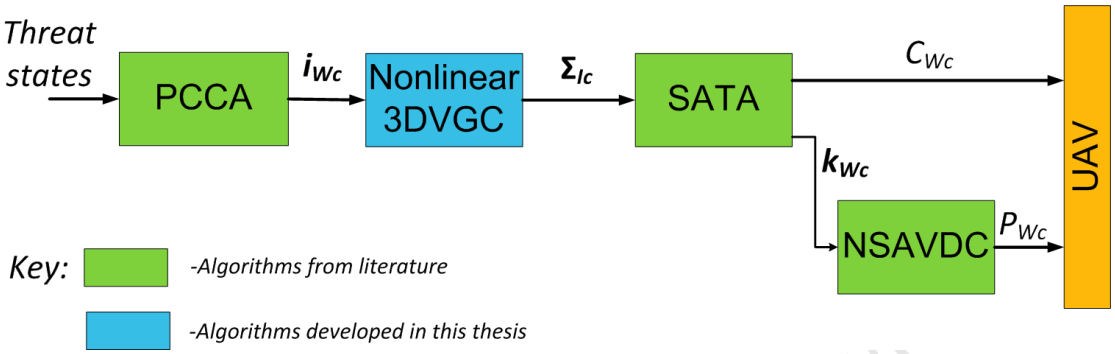

Figure 53: An overview of CASSAM V.2. The reader will note that the architecture is the same as CASSAM V.1.

In what follows, a detailed account of the design of the adapted algorithms is provided.

6.2 Projected Collision Cone Algorithm

As was previously stated, avoidance manoeuvres in the vertical direction would need to be circumvented for CASSAM V.2 as these manoeuvres are not desirable.

The Projected Collision Cone Algorithm (PCCA) achieves this by only commanding axial unit vectors which lie in the North-East plane. Thus, the UAV will only be commanded to turn and not to pitch to avoid collisions. This ties into the class of Resolution Manoeuvres of “Turns” described in the survey by Kuchar and Yang [9] described in Chapter 2.

Conceptually, the PCCA considers the Threat to be a cylinder of *infinite height*. Thus, the UAV cannot go over or under it and its only means of avoiding collision is to go around it. This concept is depicted in Figure 54.

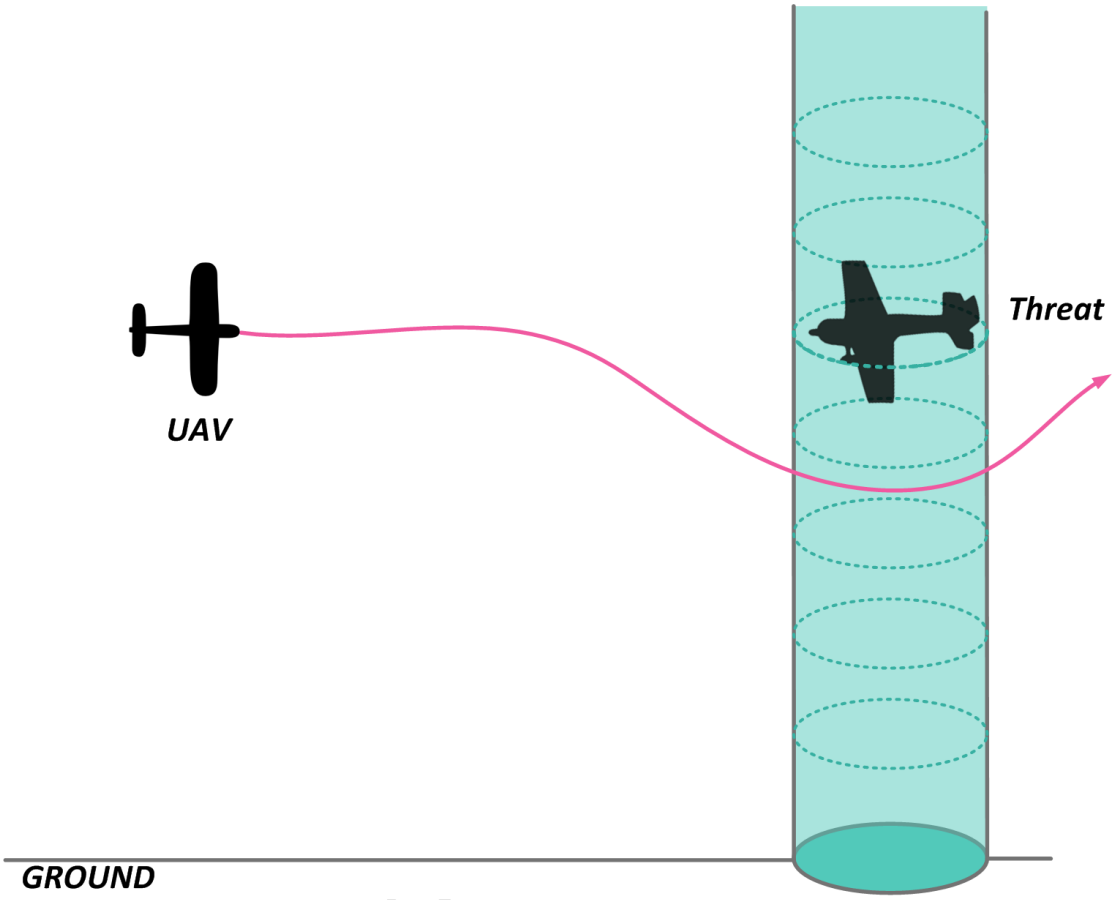

Figure 54: The PCCA perceives the threat as an infinite height cyclinder. Thus, the only way to avoid collsion is to go around the threat.

It achieves this, by projecting the and vectors, which are calculated by equations (5.19) and (5.20) respectively, into the North-East plane. This is achieved by setting the Down component in each of these to zero. This is shown as:

(6.1)

(6.2)

It was also discovered during preliminary testing that the algorithm by Watanabe [15] implemented in CASSAM V.1, did not function when threat approached from behind the UAV. The reader will recall from equations (5.1) and (5.2), the dynamic collision avoidance problem was reduced to a static one. This however, was based on the assumption that the threat would always be in front of the UAV. But the requirements from Chapter 1 require the UAV to avoid collision in an azimuth Field of Regard (FOR) ±110°. Figure 55 illustrates an approaching threat from behind. From this it is evident that by rotating the UAV's relative velocity vector will in fact rotate the *opposite* direction. This is the converse of when the threat is approaching from the front.

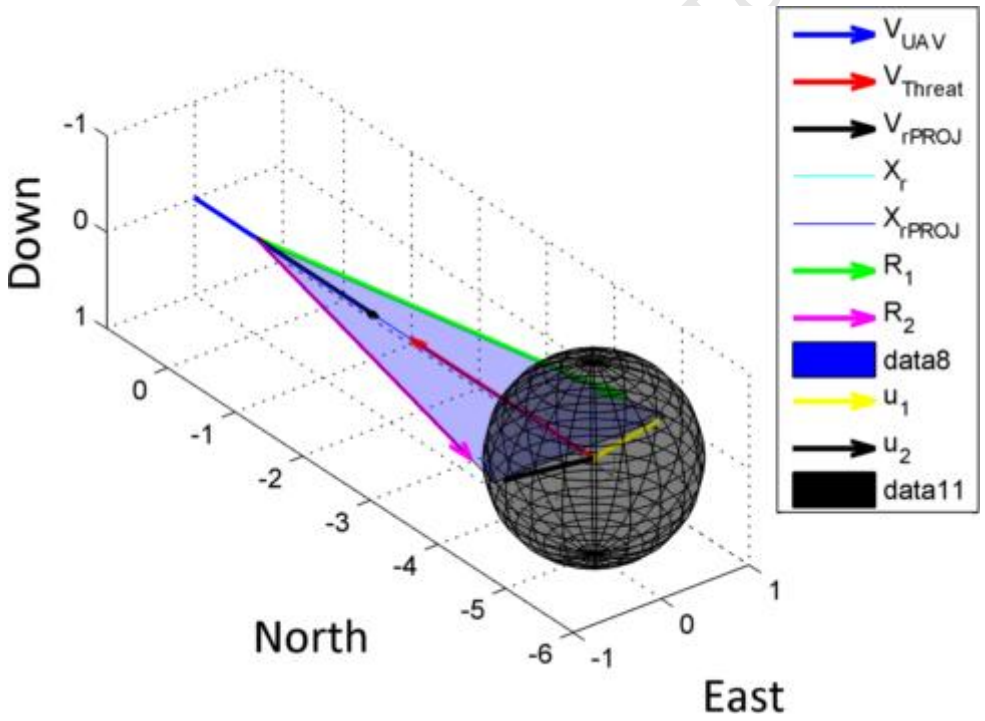

Figure 55: Threat approaching the UAV from behind. The relative velocity vector needs to be steered out of the Collision Cone.

This situation was handled by determining the angle between the UAV's velocity vector and the relative velocity vector. If the angle was greater than ninety degrees, the UAV would be commanded to rotate the opposite direction as normal. *This angle check was performed only on the first execution of the PCCA.*

Following this, the PCCA performs exactly the same algorithm as the Collision Cone Algorithm previously described in Section 5.2 to determine i_{Wc}. The vectors involved in the algorithm are depicted graphically in Figure 56.

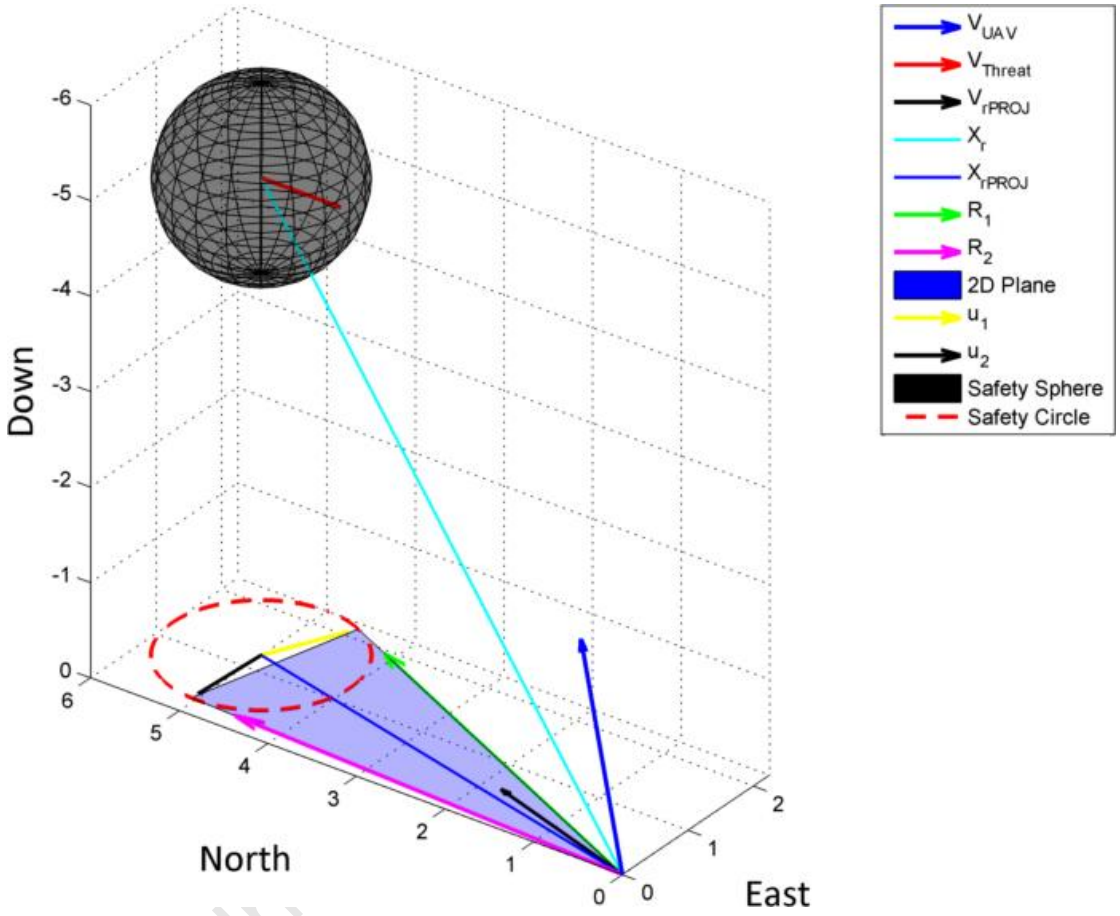

Figure 56: PCCA algorithm vectors. Note that the sphere has been projected into the N-E plane to form a circle. It is this circle that must be avoided.

During the testing of the PCCA, it was discovered that it produced false positives. It engaged collision avoidance manoeuvres even when it was involved in benign encounters. A benign encounter can be considered an encounter between another aircraft and the UAV with no risk of collision.

The reason for this can be explained by Figure 57. It is clear that these aircraft will not collide as they are separated by a sufficient vertical distance.

However, the PCCA still perceives this as an infinite cylinder it is trying to avoid and thus initiates an avoidance manoeuvre.

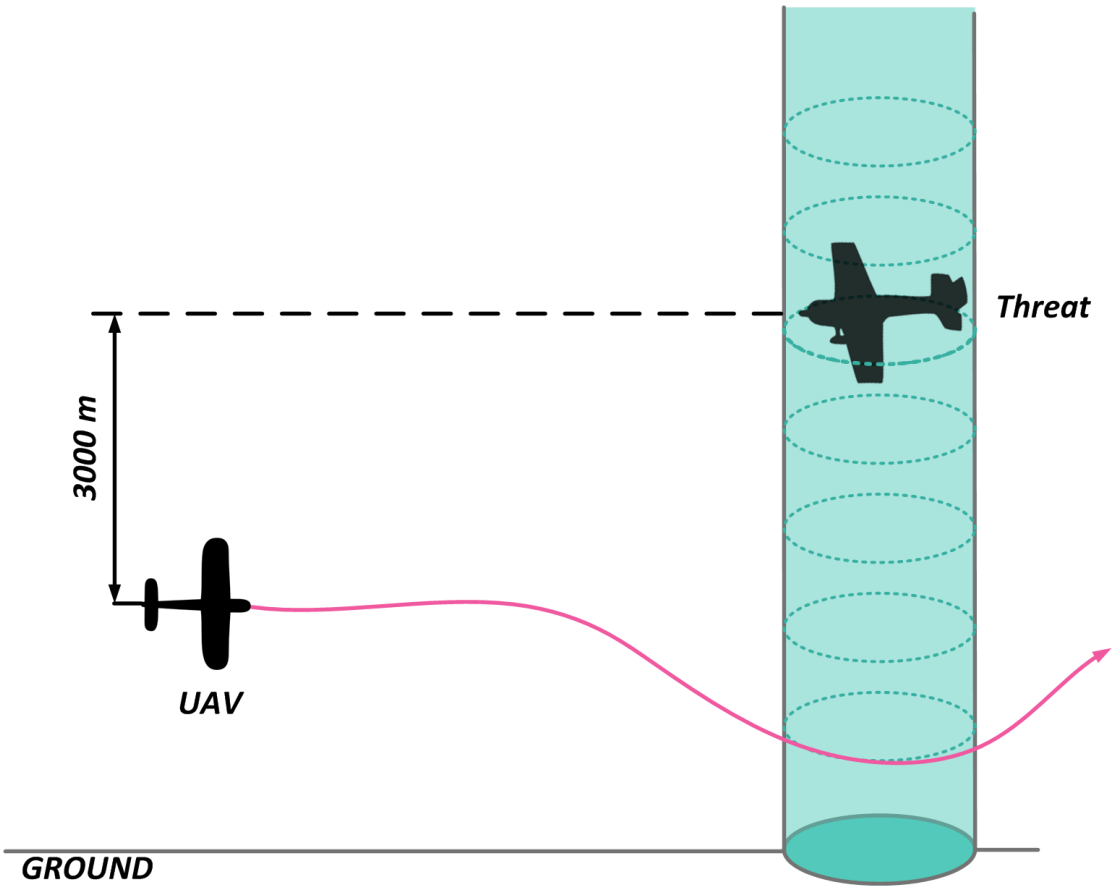

Figure 57: The false positive problem is shown. Even though the UAV and Threat are separated by a significant distance vertically (3000 m), an avoidance manoeuvre is initiated.

This problem was resolved by first performing a quick check to determine if a collision occurs. Then if this check returns as true, only then was the PCCA executed.

The Collision Check Algorithm (CCA) addresses this issue. It was implemented by employing the original “Collision Criteria” described in the original Collision Cone paper by Chakravarthy and Ghose [11]. As described in Smith [34], the criteria should be checked in the horizontal and the vertical for 3D encounters.

The CCA functions as follows. It first determines the velocity and position relative to the UAV as shown by equations (6.3) and (6.4) respectively.

(6.3)

(6.4)

These relative vectors are then converted to Spherical Coordinates as described in Appendix A.3. Finally, the Collision Criteria [11] equations are evaluated as shown below in equations (6.5) and (6.6). The reader should note that *R* is the safety radius of 150 m.

(6.5)

(6.6)

Finally, if equations (6.5) **and** (6.6) are true, only then will the PCCA be enabled.

In summary, the augmented PCCA has been described. The algorithm functions by only commanding axial unit vectors which lie in the North-East plane. Furthermore, the Collision Check Algorithm (CCA) was also implemented to prevent false positives. The PCCA will only be executed if the CCA returns true.

6.3 SAM Controllers

The SAM Controllers implemented in CASSAM V.2 are almost identical to those of CASSAM V.1. The only difference is that bank angle clipping was added to the NSAVDC. This was performed to prevent dangerous conditions

where the aircraft could tip stall or enter into a spiral dive [63]. Additionally, aircraft also have a limit on their bank angles for a particular airspeed due to the load factor increasing with the bank angle as mentioned in Section 2.5.

The bank angle is limited by calculating the bank angle from the . After the bank angle has been clipped a new unit vector will be used to determine the error angle in the NSAVDC.

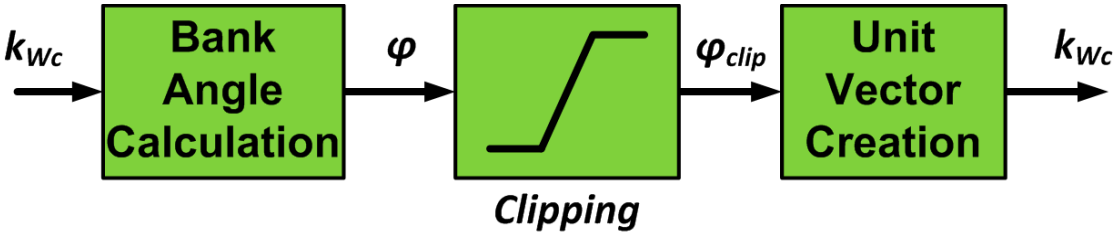

Figure 58: Overview of the process of clipping bank angle performed by the NSAVDC.

The bank angle is calculated using the equations in Section 2.4.1 and Appendix A.1.1. However, these require the ***j*** unit vector of the Wind-Axis which is not explicitly provided by the SATA. Thus, this needed to be calculated using,

(6.7)

which is derived from the fact that the Wind-Axis is a right hand and orthogonal. Also, the SATA algorithm always ensures that is perpendicular to .

Once the bank angle is determined, it is clipped to ± 35°. By using the load factor equation (2.2), it is determined that a 35° bank angle will only produce a load factor of 1.22 which is deemed as a feasible value.

Finally, employing equation (4.9), the new clipped is provided for the normal NSAVDC functionality.

The NSAVDC clipping for CASSAM V.2 has been described in this subsection. This clipping will ensure that aircraft will not enter any dangerous situations as described in Section 2.5.

6.4 Nonlinear 3DVGC

CASSAM V.1 demonstrated unexpected behaviour. Of particular importance was that the controller was much slower than anticipated. It was thus decided that a more thorough investigation into the error angle dynamics be performed to ascertain whether the assumptions made in Chapter 5 were correct.

In what follows, a detailed analysis of the error angle dynamics is documented. Following this, a new controller the Nonlinear 3DVGC is developed for CASSAM V.2 and finally, the stability of this controller is analysed by Phase Plane Analysis.

The reader should note that the Nonlinear 3DVGC employs the same Direction Vector Algorithm as the 3DVGC in CASSAM V.1.

6.4.1 Revisited Error Angle Dynamics

If the reader recalls from Chapter 5, it was assumed that the error angle dynamics were described by equation (5.26). This may have been an oversimplification and a more thorough analysis is required.

To start the new investigation, consider the UAV to be travelling at some constant total velocity ( ) depicted in Figure 59. To avoid collision, the PCCA determines that the total velocity vectors must point in a new direction

defined by the unit vector ( ). To do this, it must effectively add another component of velocity perpendicular to ( ), this will be called V^* which is depicted by the green vector in Figure 59.

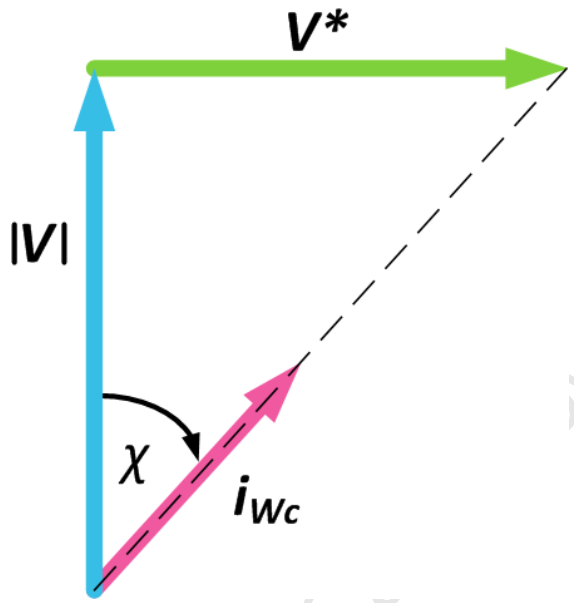

Figure 59: Error Angle Dynamics investigated further by drawing a vector diagram. The total velocity vector ( ) must be steered to point along a new direction vector ( ).

From the diagram the following relationship is evident:

— (6.8)

Then, by differentiating equation (6.8), the following equation is produced.

——— (6.9)
——

It is clear that the variable is the acceleration of the perpendicular velocity. Considering, that this always acts perpendicular to the total velocity vector, it is equivalent to the total acceleration input ( ) described in Section 5.4.2. Thus, we can consider this to be the input to the system and henceforth be known as . By substituting this into equation (6.9), the following relationship is obtained:

(6.10)

To remove the integral term this equation, equation (6.8) is altered by substituting in and simplified to obtain:

(6.11)

Finally, the error angle dynamics are shown to be:

(6.12)

It is unmistakable that when comparing this to the original error angle dynamics described by equation (5.25), that the error angle dynamics are actually nonlinear and the initial linear assumption made was incorrect. The nonlinearity arises due to the tangent function.

In light of the aforementioned information, a new control algorithm needed to be designed. This control algorithm needed to cater for the nonlinear behaviour evident in the underlying dynamics.

6.4.2 Nonlinear Control Algorithm

The nonlinear control problem should be avoided as best one can. For most applications, it is wise to make the problem seem linear by performing some mapping or linearization. Subsequently, linear methods can then be employed to design the controller as these are better understood.

Considering the error angle dynamics of equation (6.12), a new variable w is defined as:

(6.13)

Substituting this equation into equation (6.12),

(6.14)

and considering equation (5.25), the error angle becomes:

(6.15)

Equation (6.15) maps the nonlinear dynamics to that of a simple integrator with an input w. In a sense, this mapping turns the problem into a linear problem as shown in Figure 60.

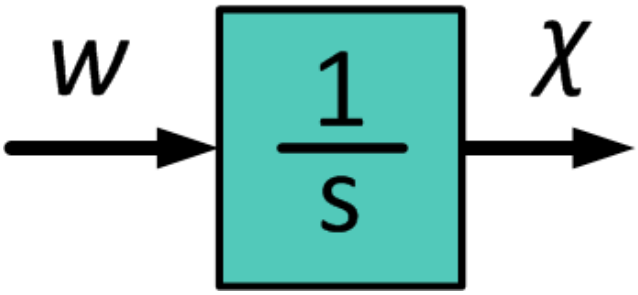

Figure 60: Linearised error angle dynamics. The nonlinear mapping greatly simplifies the system to be analysed.

This implies that the same controller described previously in Chapter 5 can be employed. This can be done because the mapping of equation (6.13) will take care of the nonlinear terms. Thus, the controller is given as:

(6.16)

By placing the poles in the same position as the original 3DVGC, the controller becomes:

(6.17)

However it should be noted that this controller outputs a w command which still needs to be mapped back to nonlinear system. Thus, the output of this controller still needs to be mapped back to . This is performed by utilizing the following relationship:

(6.18)

It is evident that the mapping will output an infinite acceleration at 90°. However, this will only occur when the threat and the UAV are occupying the same space (collision) and thus, this is deemed to be an unlikely event.

The block diagram of the entire control architecture is shown below in Figure 61.

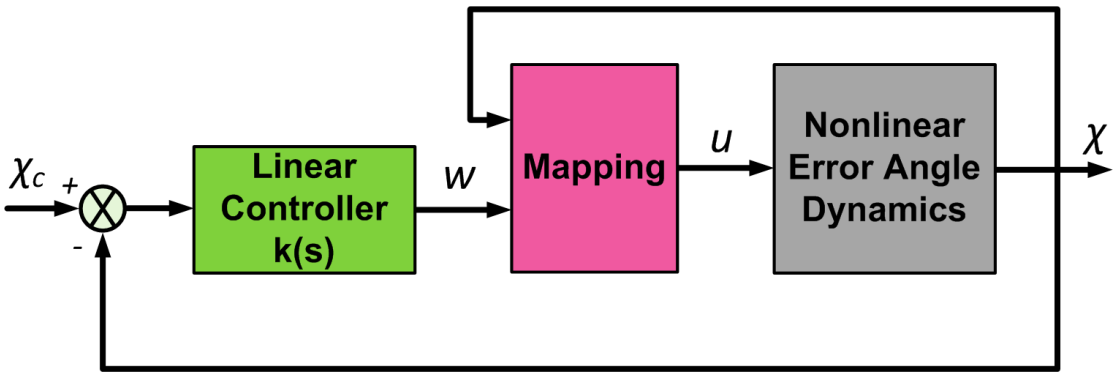

Figure 61: Block diagram of the Nonlinear Controller for CASSAM V.2. The total controller is comprised of a linear control algorithm and nonlinear mapping.

A final note is that because this controller is used in conjunction with the DVA described in Chapter 5, the sign of must always be positive. This is because the DVA will always point the total acceleration vector in the correct direction. Thus, the absolute value of was utilized when the algorithm was integrated into CASSAM V.2.

In this subsection, the new Nonlinear 3DVGC was presented and designed. Unfortunately, the error angle dynamics have been shown to be nonlinear and thus, linear methods cannot be used to analyse the system behaviour. The next subsection will describe a method for analysis of nonlinear systems to illustrate the stability and performance of this controller.

6.4.3 *Phase Plane Analysis*

Phase Plane Analysis [64] is a useful method for analysing, *second order autonomous* systems. The Phase Portrait is a graphical depiction of the trajectories of how the states in a system evolve over time. Example phase portraits are shown in Figure 62.

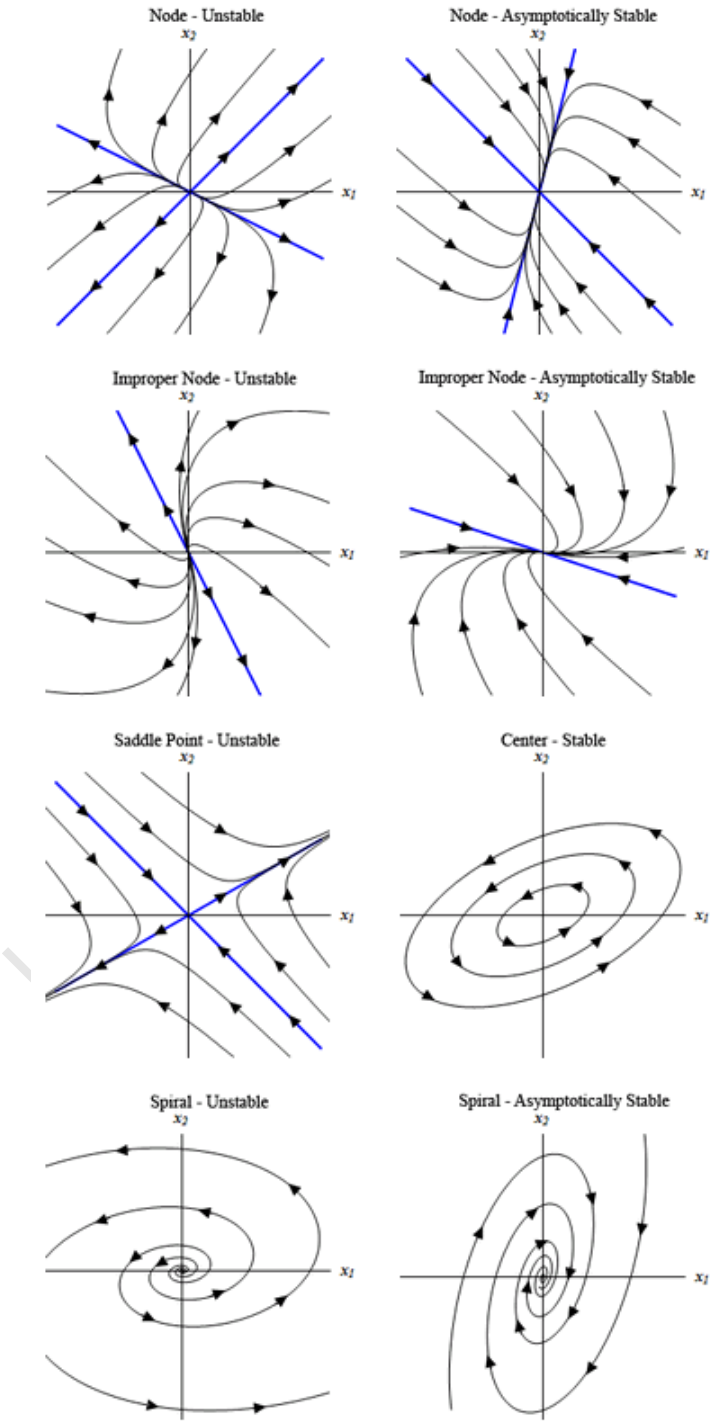

Figure 62: Example Phase Portraits for different systems [65].

The reader should notice the difference between **stable** and **asymptotically stable**. Asymptotically stable nodes or spiral trajectories will move in towards the equilibrium point as time progress. A centre (which is always stable) trajectories will just move around the equilibrium point but never actually move in towards it [65].

By employing the error angle dynamics of equation (6.12), the trajectory of the state ( ) can be plotted. The reader will also recall the use of a PI control algorithm, which by virtue of its integral action, added another state to the system. This will require the state vector to be augmented as is described in [66]. Thus, the new state is added and is defined as:

(6.19)

where, *e* is the error signal.

With reference to the method described in [66], the control architecture for this system is illustrated in Figure 63.

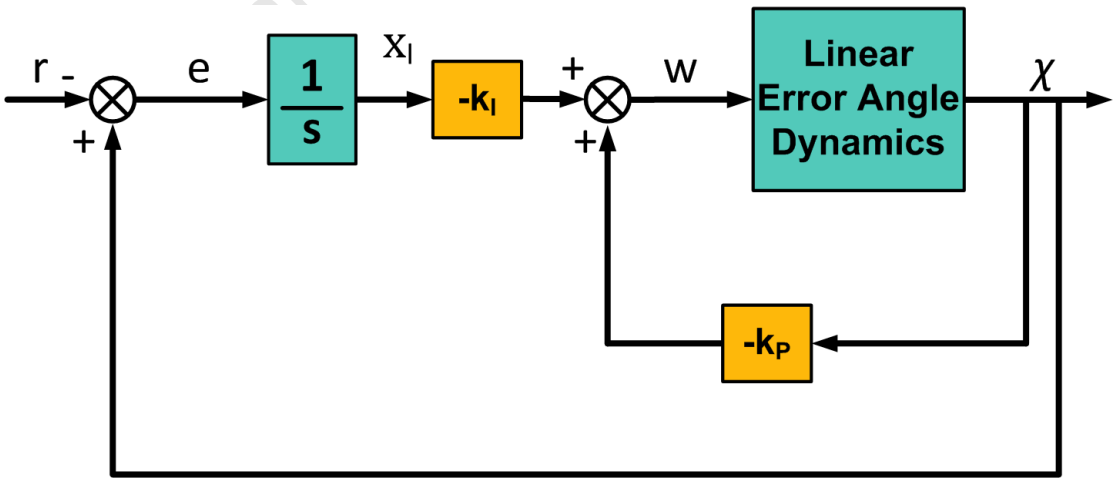

Figure 63: Integral control structure in state space. Note that the linear dynamics containing the nonlinear mapping w.

The state equations are then shown to be:

(6.20)

(6.21)

The reader should note that this is then mapped to the linear equivalent model by equation (6.13) previously stated.

Furthermore, the linear control transfer function is provided by equation (6.17) which results in the state space control equation:

(6.22)

This is mapped back by utilizing equation (6.18) and thus, the total nonlinear control law is shown to be:

(6.23)

This is substituted into equation (6.21) to provide an autonomous system [64].

Finally, equations (6.20) and (6.21) are numerically integrated using Matlab at a total velocity of 35 m/s. The resulting phase portrait is depicted in Figure 64. From this figure it is evident that the system is stable because the starting the states always converge to zero. Figure 65 also demonstrates how the states converge with time. It is evident that the controller forces the states to converge with a constant settling time in varying initial conditions.

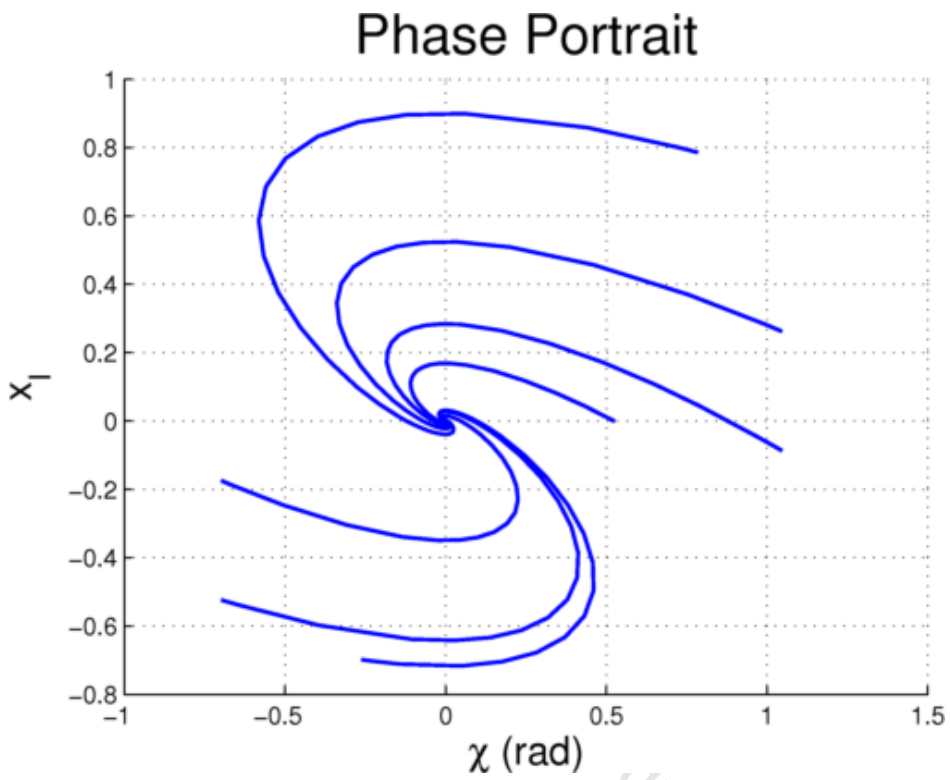

Figure 64: Phase Portrait of the Nonlinear 3DVGC employed in CASSAM V.2. Note that the system is asymptotically stable.

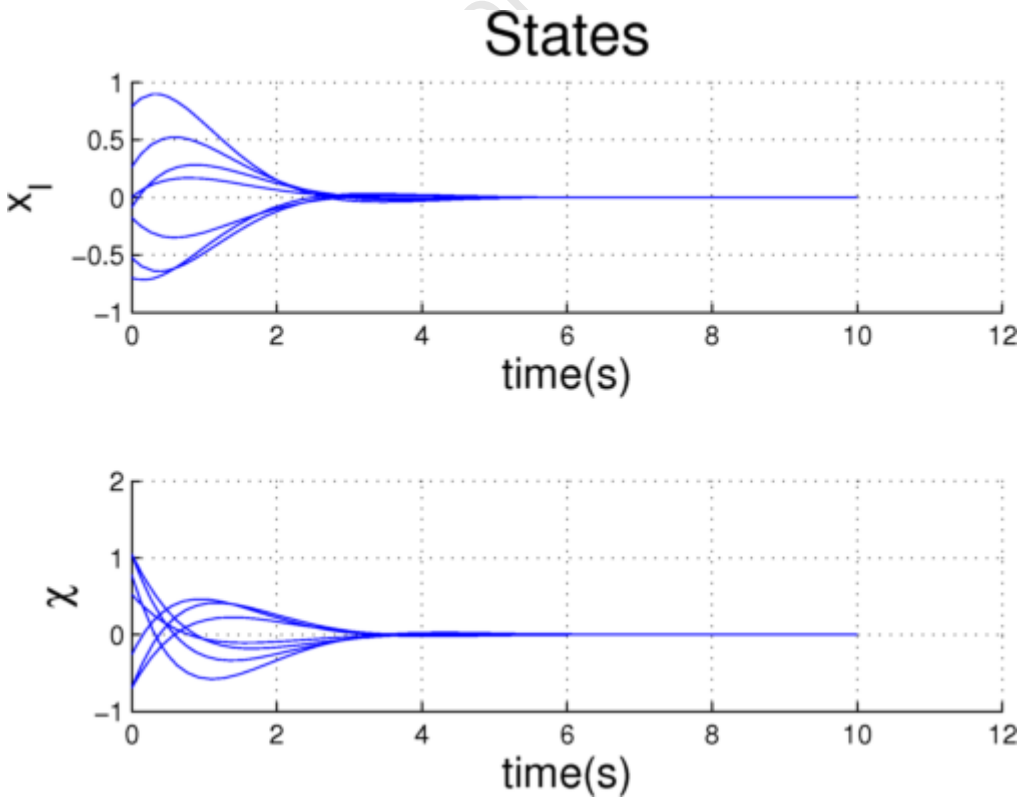

Figure 65: State trajectories with respect to time. The states settle with a constant settling time for different initial conditions.

It was also desired to plot the state trajectories of the system under the control of the linear 3DVGC designed in CASSAM V.1. The reader will recall that the design of this controller was made under an assumption of linearity, which was later proved to be incorrect.

Figure 66 and Figure 67 illustrate that this controller did not behave adequately. Though it did not introduce stability, the states took a considerable time to converge.

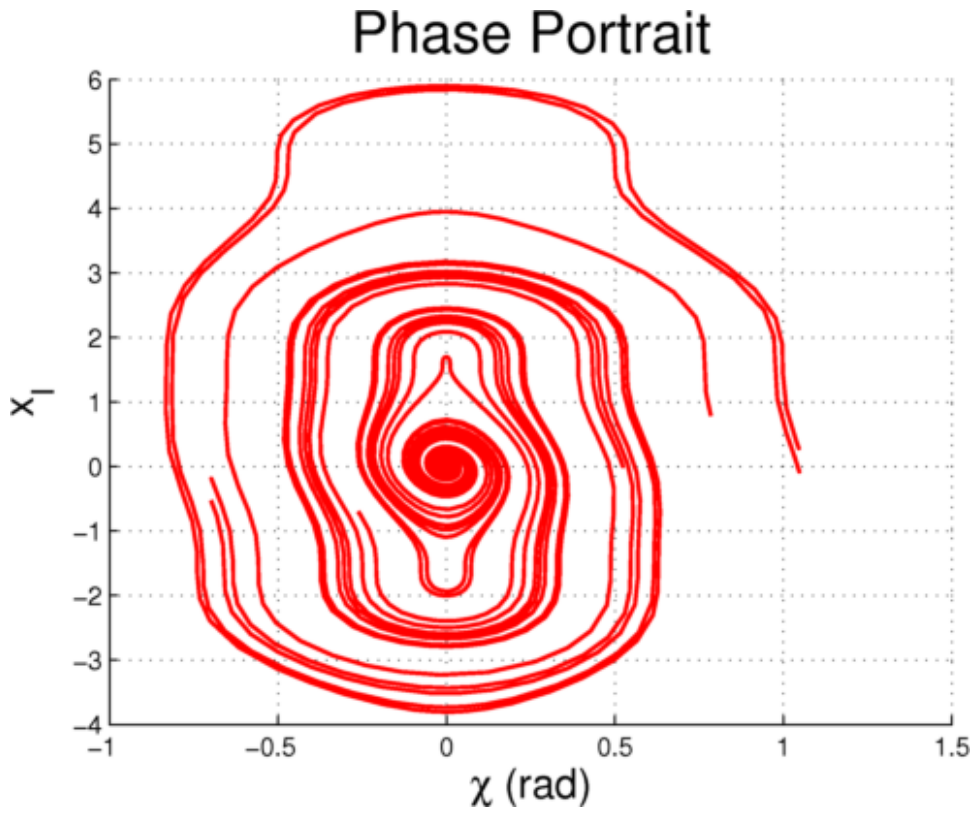

Figure 66: Phase Portrait of the linear 3DVGC employed in CASSAM V.1.

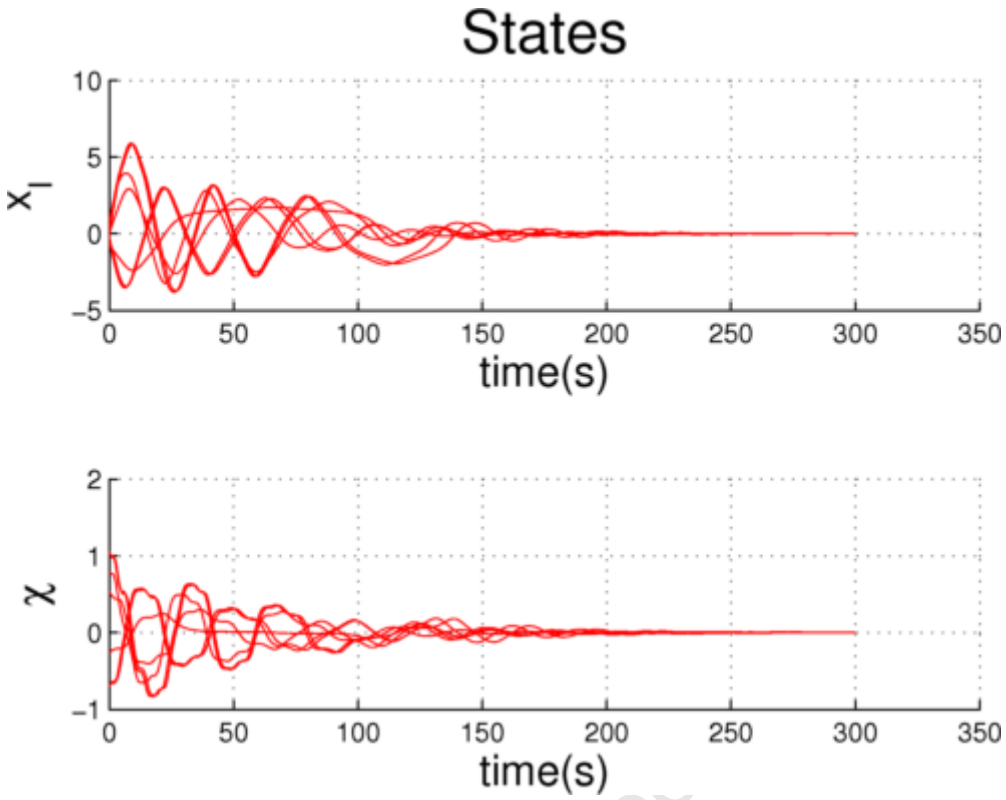

Figure 67: State trajectories with respect to time. The states settle with different settling times for different initial conditions.

It is evident from the Phase Portraits for the respective control algorithms that the Nonlinear 3DVGC is the superior algorithm. This is primarily attributed to the incorrect assumption of the error angle dynamics made in Chapter in 5. In the next subsection, the Nonlinear 3DVGC was integrated into CASSAM V.2 and tested in the full simulation environment.

6.5 Simulation Results

The aforementioned algorithms were then implemented in Simulink and integrated into the main simulation described in Chapter 4. CASSAM V.2 was then tested under the discrete scenarios described in the Methodology and the results are depicted below.

6.5.1 *Head on Collision (S0) - TTI 20s*

The initial states are depicted in Table 6. The reader should note that all vectors are in the NED frame.

Table 6: Initial States for S0 with TTI 20s for CASSAM V.2.

Initial States	Velocity (m/s)	Position (m)	Heading (°)
UAV	[35 0 0]	[0 0 0]	0
Threat	[-100 0 2]	[2700 0 – 40]	180

The resulting trajectory plot is illustrated by Figure 68. Note that the blue dot and red sphere depict the positions of the UAV and threat at t(CPA). It is shown that the UAV manoeuvres to the right and adjusts its heading to successfully avoid the collision.

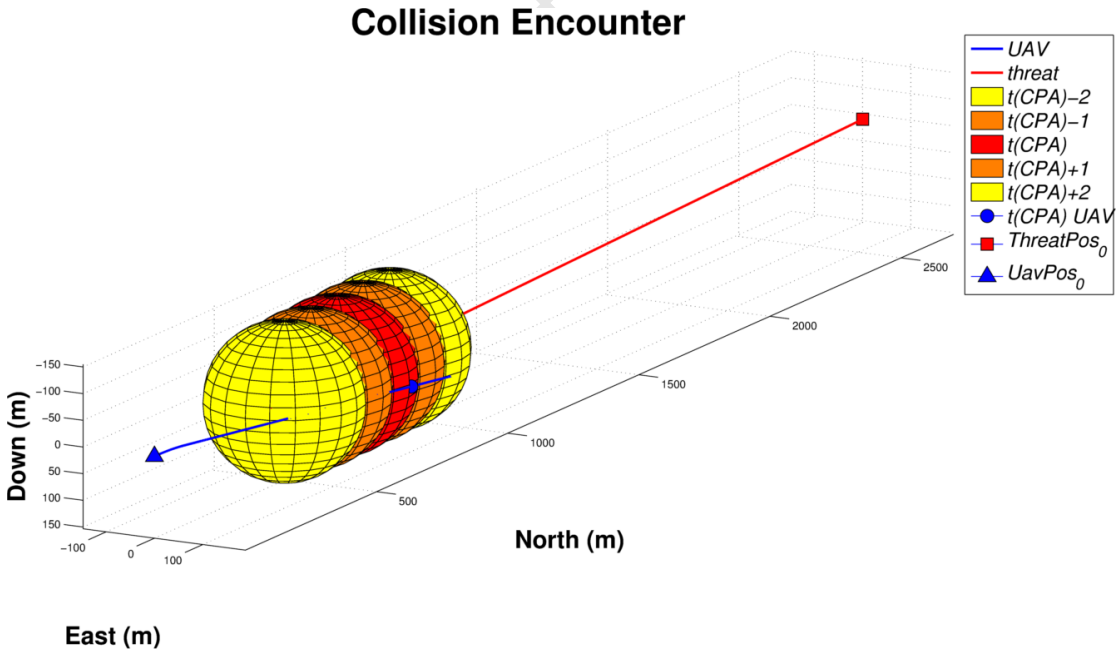

Figure 68: Trajectory plot for S0 with TTI 20s for CASSAM V.2. It is evident that the threat has been avoided.

The data captured is graphically illustrated in Figure 69. For this particular scenario, a maximum bank angle of 26.59° during the evasive manoeuvre.

The UAV experienced a peak NSA of -10.93 m/s^2. It was also noticed that the UAV pitch angle was negligible during the effective evasion of the threat.

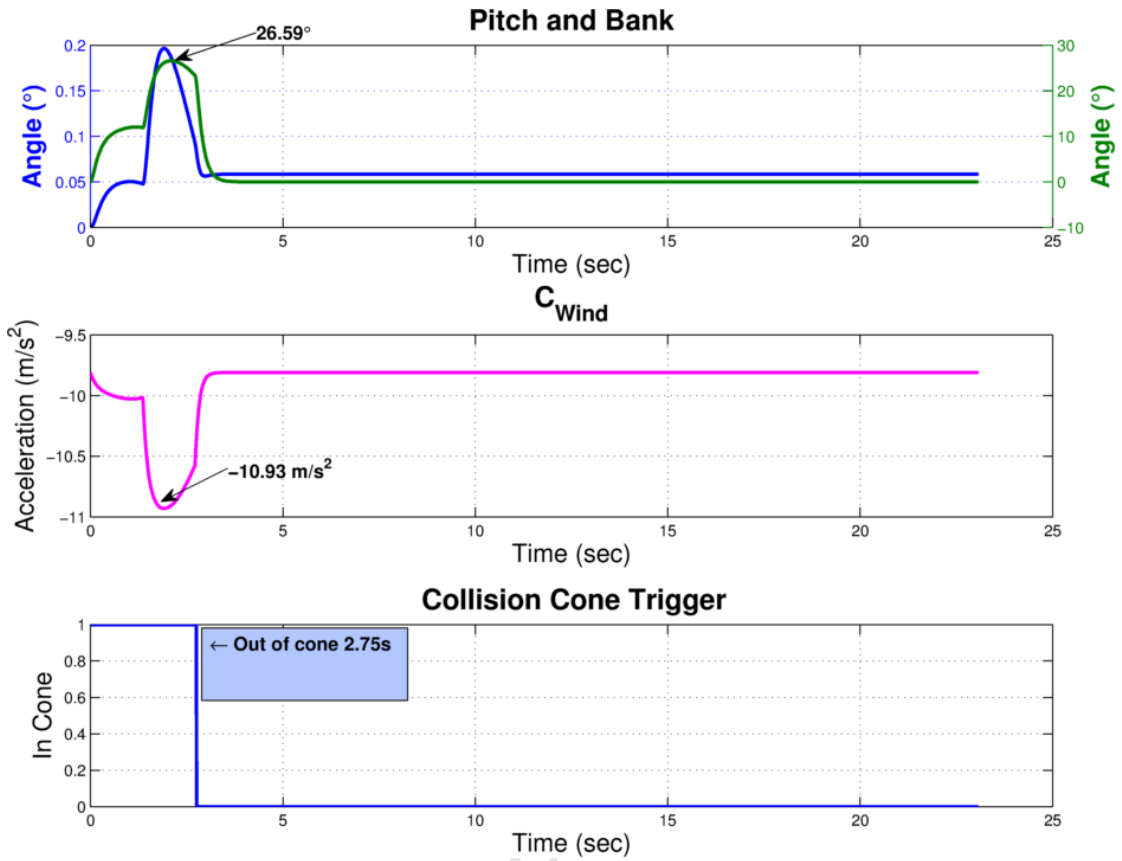

Figure 69: Data captured for S0 with TTI 20s for CASSAM V.2. The Time-to-Safety was 2.75s

6.5.2 Side on Collision (S1) - TTI 20s

The initial states are depicted in Table 7. The reader should note that all vectors are in the NED frame.

Table 7: Initial States for S1 with TTI 20s for CASSAM V.2.

Initial States	Velocity (m/s)	Position (m)	Heading (°)
UAV	[35 0 0]	[0 0 0]	0
Threat	[0 -120 -2]	[700 2400 40]	-90

The trajectory plot for the encounter is depicted by Figure 70. The plot shows that UAV banks to the right to successfully avoid collision.

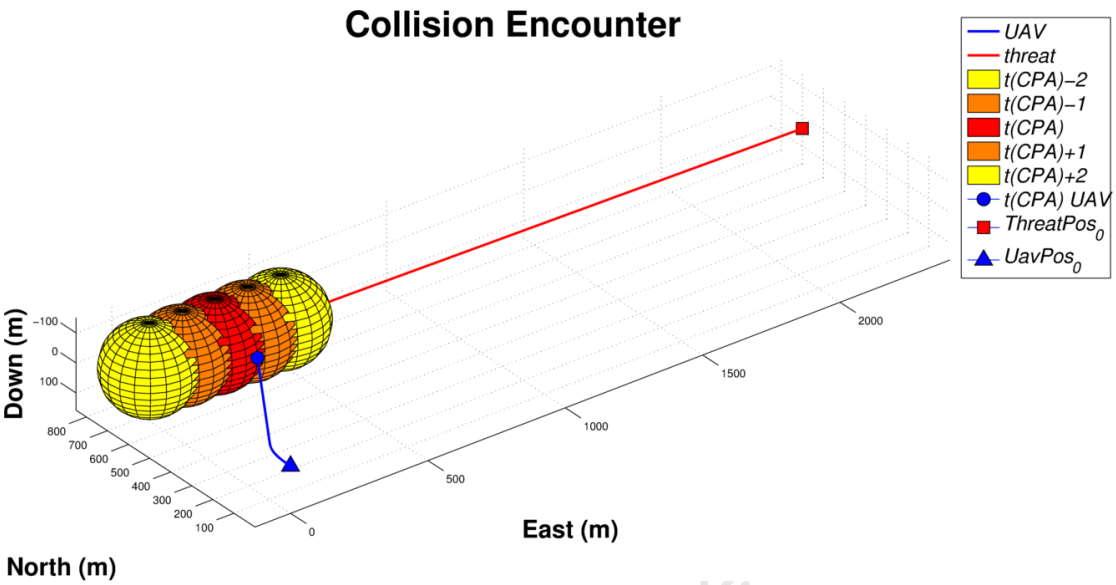

Figure 70: Trajectory plot for S1 with TTI 20s for CASSAM V.2. It is evident that the threat has been avoided.

The data captured for this scenario is plotted in Figure 71. The UAV successfully evades the threat by commanding a maximum bank angle of 35.72°. This bank angle was clipped by the NSAVDC of CASSAM V.2 as indicated by the plot. The maximum NSA commanded was -12.6 m/s^2 and a negligible pitch angle of 1° was experienced to achieve this. The time to safety was found to be 3.94s.

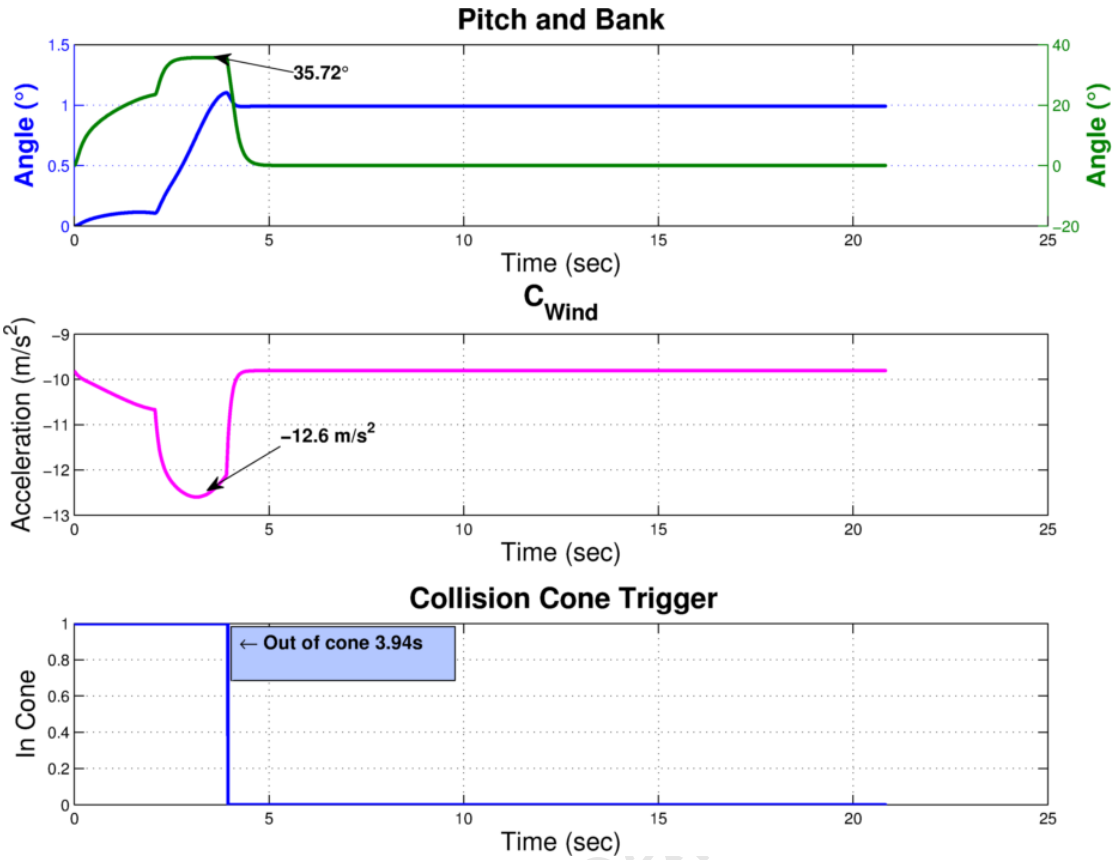

Figure 71: Data captured for S1 with TTI 20s for CASSAM V.2. The Time-to-Safety was 3.94s

6.5.3 Head on Collision (S0) - TTI 10s

The initial states are depicted in Table 8. The reader should note that all vectors are in the NED frame.

Table 8: Initial States for S0 with TTI 10s for CASSAM V.2.

Initial States	Velocity (m/s)	Position (m)	Heading (°)
UAV	[35 0 0]	[0 0 0]	0
Threat	[-100 0 2]	[1350 0 -20]	180

The trajectory plot is depicted by Figure 72. It is evident that the UAV has successfully evades the threat by banking to the right.

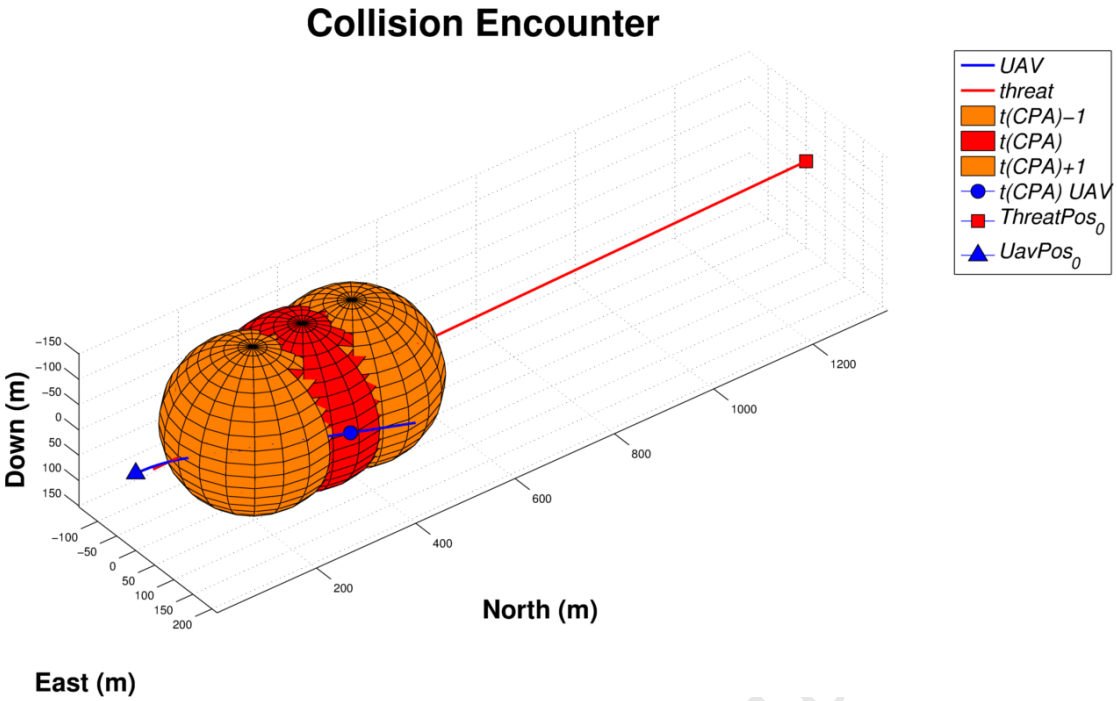

Figure 72: Trajectory plot for S0 with TTI 10s for CASSAM V.2. It is evident that the threat has been avoided.

The data captured is plotted in Figure 73. As in the previous example, the NSAVDC attempted to clip the bank angle and the maximum angle experienced was 36°. In this scenario, the NSA command was clipped by the NSA virtual actuator to -14.72 (1.5 times gravity). The pitch angle commanded to achieve this acceleration was 5°. The reader will notice that this pitch angle is held after the CASSAM disengages. Additionally, the TTS was determined to be 2.38s.

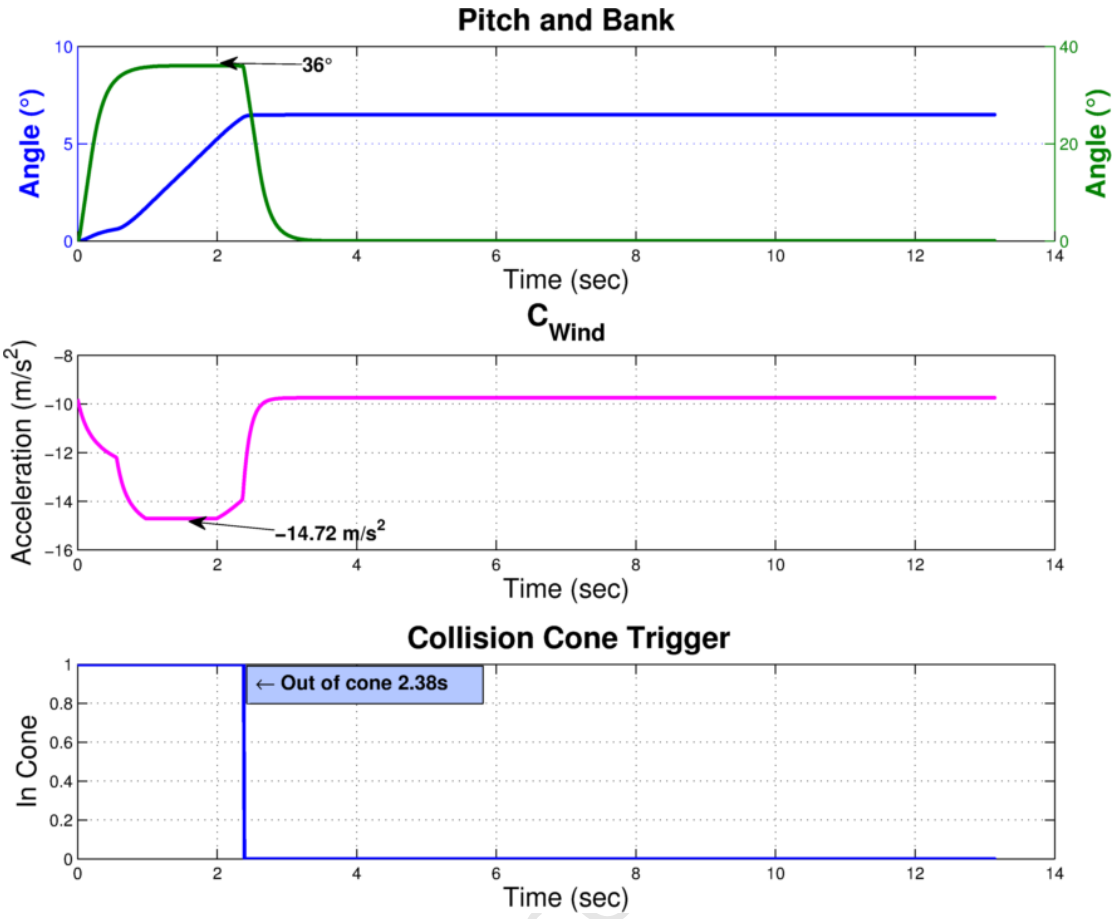

Figure 73: Data captured for S0 with TTI 10s for CASSAM V.2. The Time-to-Safety was 2.38s

6.5.4 Side on Collision (S1) - TTI 10s

The initial states are depicted in Table 9. The reader should note that all vectors are in the NED frame.

Table 9: Initial States for S1 with TTI 10s for CASSAM V.2.

Initial States	Velocity (m/s)	Position (m)	Heading (°)
UAV	[35 0 0]	[0 0 0]	0
Threat	[0 -120 -2]	[350 1200 20]	-90

The trajectory plot is graphically illustrated by Figure 74. It can be seen that the UAV successfully avoids collision by evasively banking to the right.

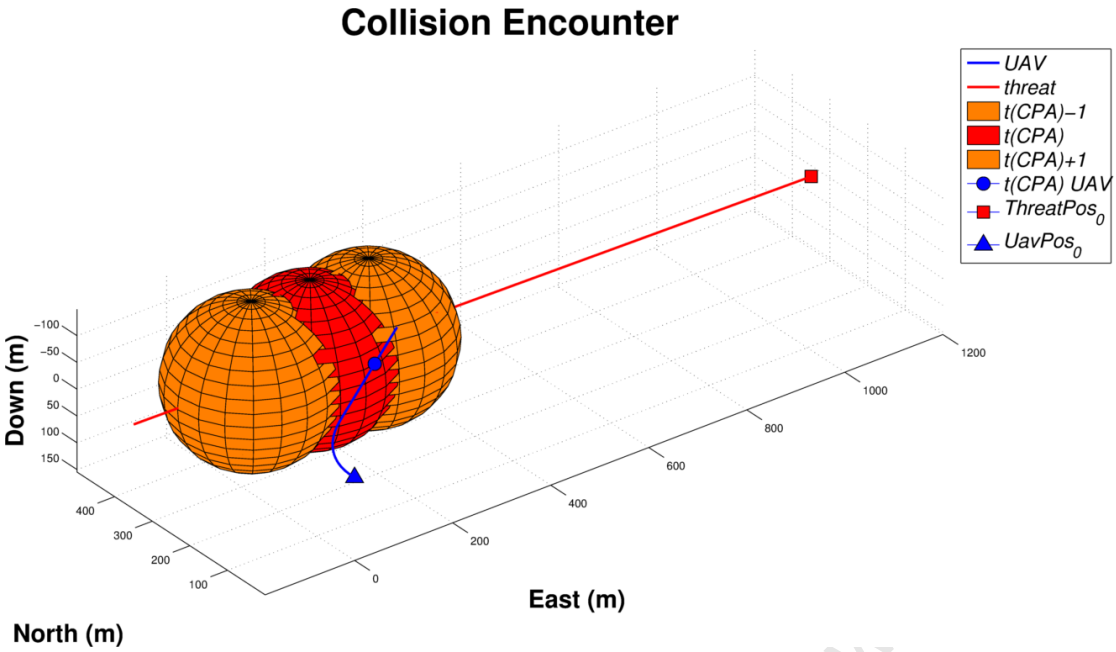

Figure 74: Trajectory plot for S1 with TTI 10s for CASSAM V.2. It is evident that the threat has been avoided.

The data obtained from this scenario is plotted in Figure 75. The UAV achieved a maximum bank angle of 36° which was clipped by the NSAVDC. In addition to this, CASSAM V.2 commanded an NSA of -14.72 which was clipped by the underlying NSA virtual actuator. To achieve this magnitude of acceleration (and sustain it), the UAV's pitch angle reached a maximum of 15.9°. This value was held constant after CASSAM V.2 was disengaged. This meant that the UAV pulled up slightly in the turn. Furthermore, the TTS was 4.45s, which was larger than S1 at 20s.

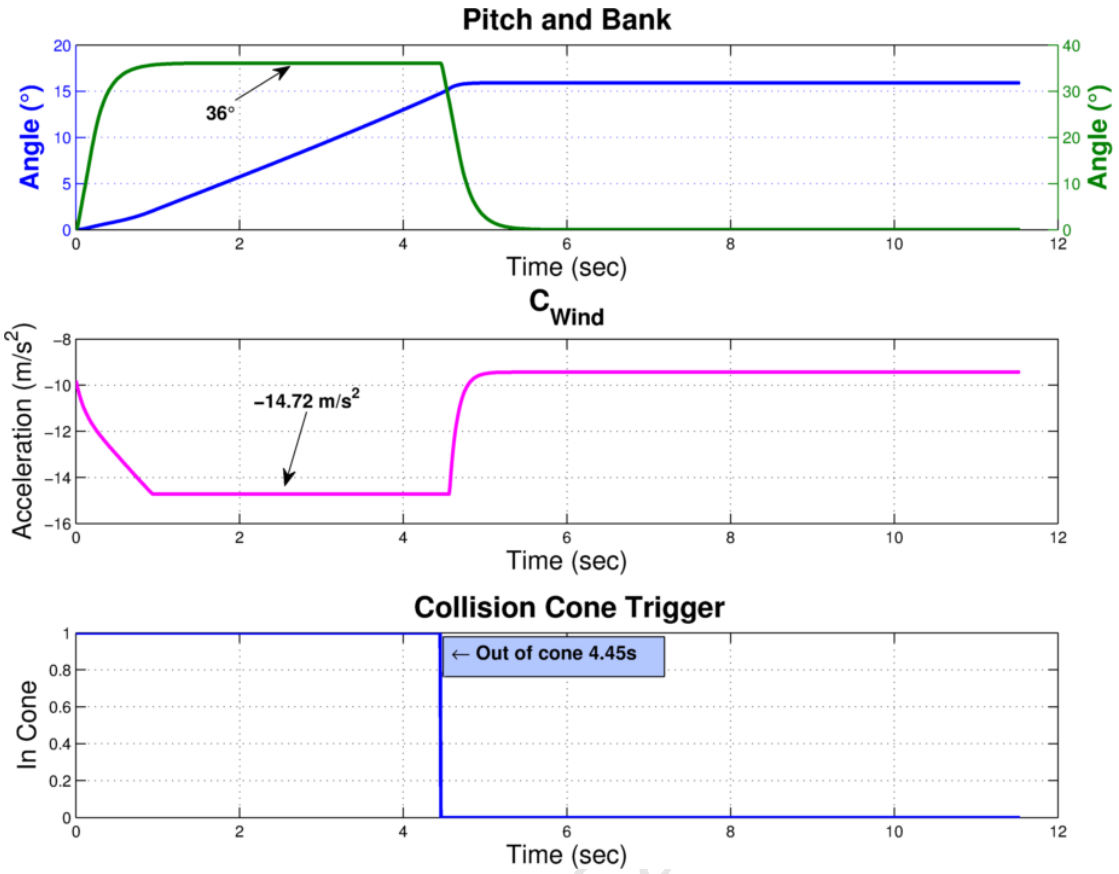

Figure 75: Data captured for S1 with TTI 10s for CASSAM V.2. The Time-to-Safety was 4.45s

6.5.5 Summary

In summation, CASSAM V.2 successfully evaded collision in all encounter scenarios tested. It achieved this by banking to turn and guide the UAV out of the Collision Cone.

The data illustrates that the clipping functions (bank and NSA) are behaving as expected. Additionally, it was shown that during S1 for TTI 10s that the pitch angle also increased sympathetically with the NSA commanded. Furthermore, when CASSAM V.2 disengaged, the pitch angle was held constant. Additionally, it was shown that during S1 for TTI 10s, the TTS was 4.45s.

Chapter 7: Monte Carlo Simulation & Analysis

As discussed in Chapter 2, Monte Carlo simulation methods would be utilized to test the collision avoidance system. Monte Carlo simulations allow one to simulate the element of uncertainty inherent to the operating environment. In addition to this, the Monte Carlo simulation will also provide a handle on the robustness of the collision avoidance system in varying scenarios. Finally, using the data obtained from the simulations, statistical analysis can be performed and conclusions can be drawn. In what follows, Monte Carlo Simulation was performed and analysis was performed on the data generated.

The reader will recall from Chapter 5 that CASSAM V.1 consistently produced dangerous manoeuvres to evade collisions. In a practical scenario this system would not be implemented for safety reasons. Thus, *due to the unsatisfactory behaviour of CASSAM V.1, only CASSAM V.2 would be tested by Monte Carlo simulation.*

7.1 Overview of Simulation

For this thesis, the random input to the system was the encounter scenario. Thus, many random encounters needed to be generated for a selection of Times-to-Impact (TTI).

For each TTI, one thousand simulation runs were executed. The main aspects noted are depicted in

Table 10, which is an example of the table to be utilized for data capture. The reader should note that

Table 10 does not contain any data.

Table 10: Monte Carlo Simulation parameters for each TTI.

TTI (s)	25	20	15	12	10	8	6
NMACs							
Threat States							
TTS							

In order to obtain this data, the simulation environment developed in Chapter 4 was appended with two more functional blocks. These were:

- The Random Encounter Algorithm
- The Collision Monitor

A functional overview of these is depicted graphically in Figure 76.

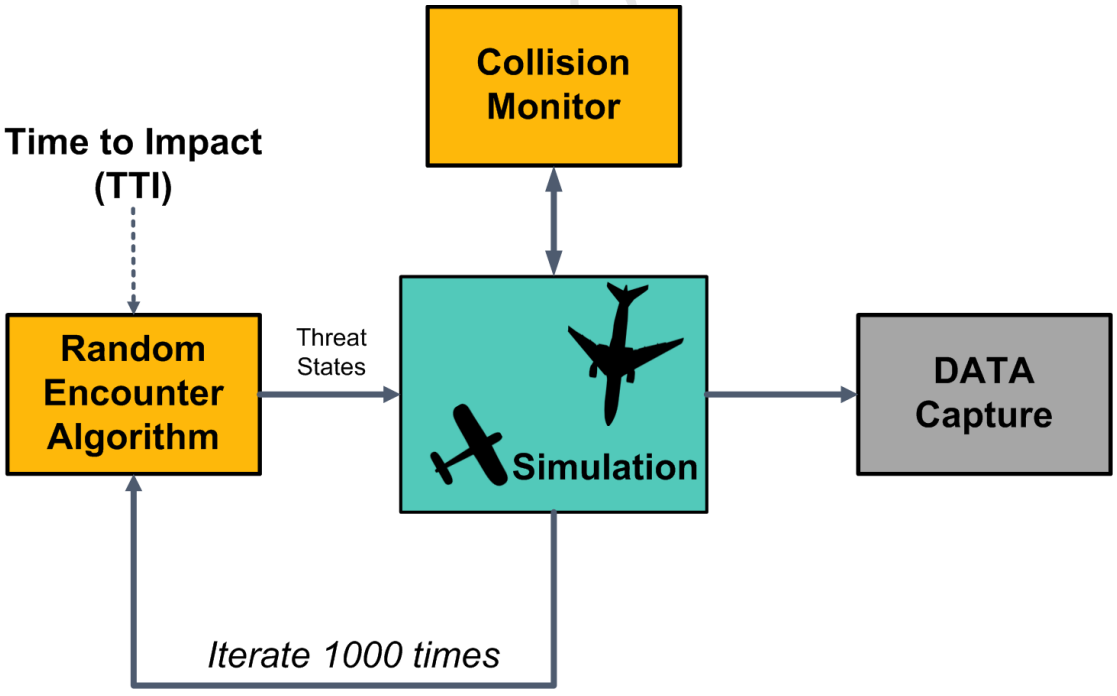

Figure 76: Overview of the Monte Carlo Simulation architecture.

To perform the Monte Carlo simulations (with reference to Figure 76), the process is as follows:

- A random encounter algorithm generates threat position and velocity states.
- These states are then fed into the main simulation.
- The collision monitor algorithm observes the simulation and if a collision occurs or if it is evaded, it halts the simulation.
- The simulation data is saved and the process repeats itself one thousand times.

A simulation script was created to repetitively execute the simulation and save the data for post-processing and analysis.

In what follows, the development of the additional algorithms is described.

7.2 Random Encounter Algorithm

In order to efficiently utilize the simulation time, only valid threat encounters were generated. Thus, only threat states (positions and velocities) which would result in collision were generated.

To achieve this, the Random Encounter Algorithm was created. Fundamentally, the algorithm determines random position and velocity vectors for the threat at a given TTI. A flow diagram depicting the algorithm is illustrated in Figure 77.

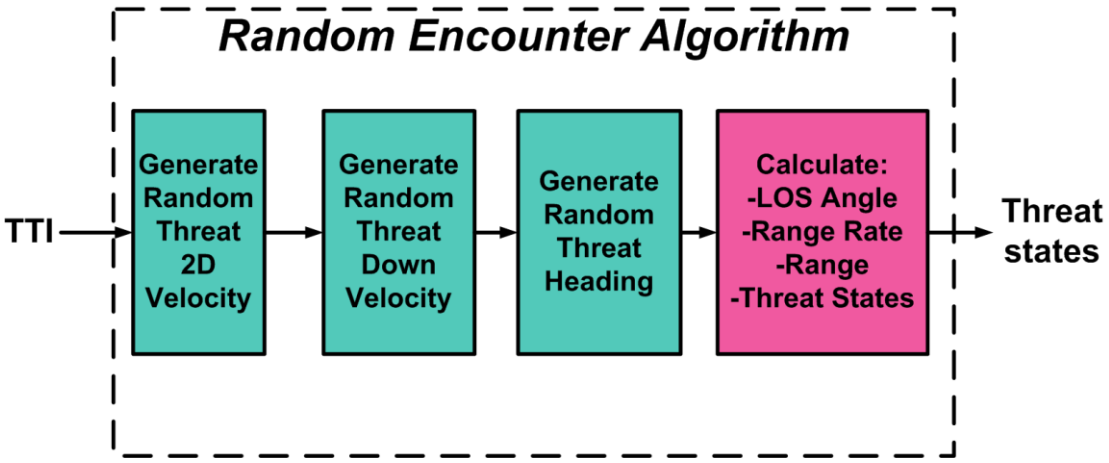

Figure 77: Outline of Random Threat Generation algorithm.

On first execution, the algorithm generates a random threat velocity. This velocity is considered to be a uniform random variable on an interval between 40 m/s to 150 m/s. This is to emulate the airspeeds of various VFR traffic. As discussed in Chapter 1, VFR is where most NMACs occur.

Subsequent to this, a random Down velocity component for the threat is generated. Most aircraft generally do not climb or descend abruptly as this will produce excessive load factors. Thus, the interval on which Down velocity can be generated is between -10 m/s to 10 m/s.

Then, a random threat heading is generated. To satisfy the field of regard requirement discuss in Chapter 1, the interval is selected to be 30° to 330° because these approximately produced threats with initial azimuth angles of ±110°.

The final stage of the algorithm is the calculation section. To calculate a relative azimuth or Line of Sight (LOS) angle which will result in collision, Lemma 3 of [11] is employed. For the sake of completeness, this is shown by equation (7.1).

(7.1)

Where, and are the UAV and threat headings respectively. Using this, the range rate or can be calculated. This is also provided by [11] and is repeated here:

(7.2)

Now that has been calculated and the TTI has been specified, the range can be calculated using equation (7.3). The absolute value is used to remove the negative sign of .

(7.3)

Next, to determine the Down component of the position vector, the following equation is applied:

(7.4)

The reader will notice that we now have all the elements of a Cylindrical Coordinate system (See Appendix A.2). It is then a trivial task to convert these to the Cartesian (NED) frame.

This algorithm was coded in Matlab and called by the Monte Carlo script.

7.3 Collision Monitor

The Collision Monitor checks the simulation and performs two primary functions:

- Check that the threat has been evaded.
- If the threat has not been evaded, stop the simulation.

To check whether the threat was indeed evaded, the Collision Monitor calculates the line of sight rate ( ). If the line of sight rate increases before collision occurs, the closest point of approach (CPA) is considered to have occurred. Thus, collision has been avoided and the simulation is stopped.

Simultaneously, the Collision Monitor also calculates the line of sight distance . If this distance is smaller than the safety radius (150 m), an NMAC is considered to have occurred and the simulation is stopped.

The data output is logged and was used in the Monte Carlo analysis. The reader should note that this algorithm utilizes the Spherical Coordinate definitions described in Appendix A.3.

This algorithm was implemented in a Simulink S-function and formed part of the Monte Carlo simulation.

7.4 Simulation Results

The results of the Monte Carlo simulation for CASSAM V.2 are depicted here. The data plots were generated using Matlab to aid visualization and interpretation of the data.

7.4.1 Probability of Near Mid-Air Collision – P(NMAC)

This data was derived by considering all the NMACs occurred and by use of equation (3.1) described in Chapter 3. This was performed for each TTI and is illustrated graphically by Figure 78.

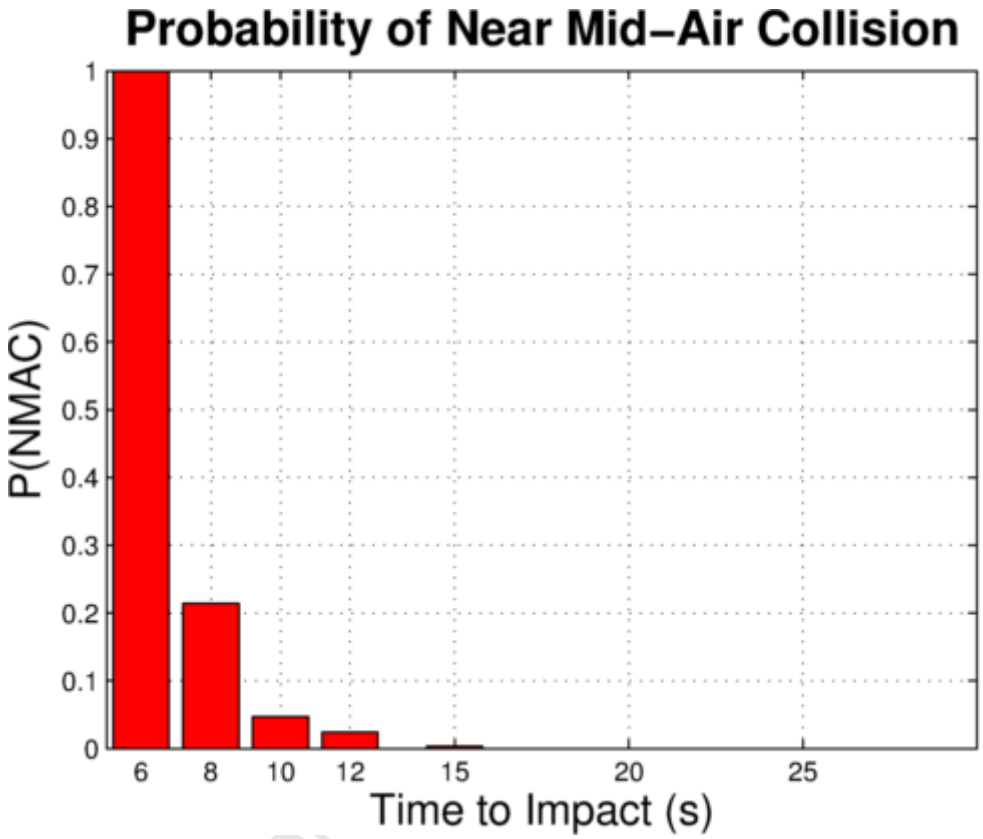

Figure 78: Probability of Near Mid-Air Collision for various TTIs.

As shown in Figure 78, a time to impact of 6s always lead to a collision. However a time to impact of over 12s provided a probability of less than 0.01 for a collision.

7.4.2 Spatial Distribution of NMACs

Another way to evaluate the success of a CAS is to investigate the parameters of distance between the UAV and threat, together with the velocities of these aircraft. Therefore a plot of distance between UAV and threat with their velocities will be done. For this plot data from the TTI 8s

simulation was plotted as it provided a sufficient contrast between scenarios which were evaded and those which were not. However to show these three-dimensional properties, a three-dimensional scatter plot has been generated. The velocity vectors and their initial positions are plotted in Figure 79.

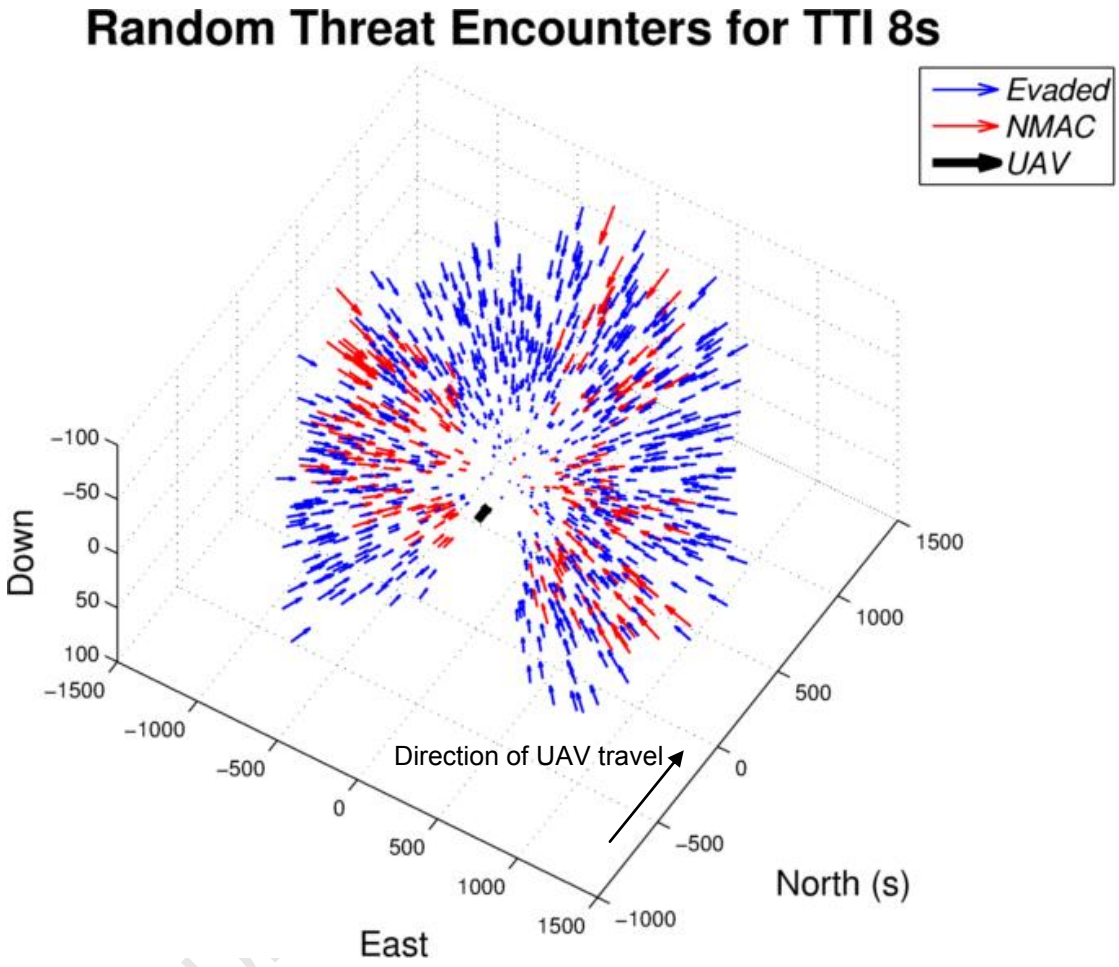

Figure 79: Spatial distribution of NMACs for TTI 8s. The blue arrows represent the threats avoided and the red arrows represent those which caused a NMAC.

This plot demonstrates that the majority of NMACs (shown by red arrows) started out from a position on one of the sides of the UAV. There is also a lesser amount of NMACs occurring in the head-on scenario.

To further investigate why most NMACs occurred from the sides, a Rose Plot[12] has been used to ascertain any angular position correlation to NMACs. This is illustrated by Figure 80. This plot uses all TTIs data (except for 6s).

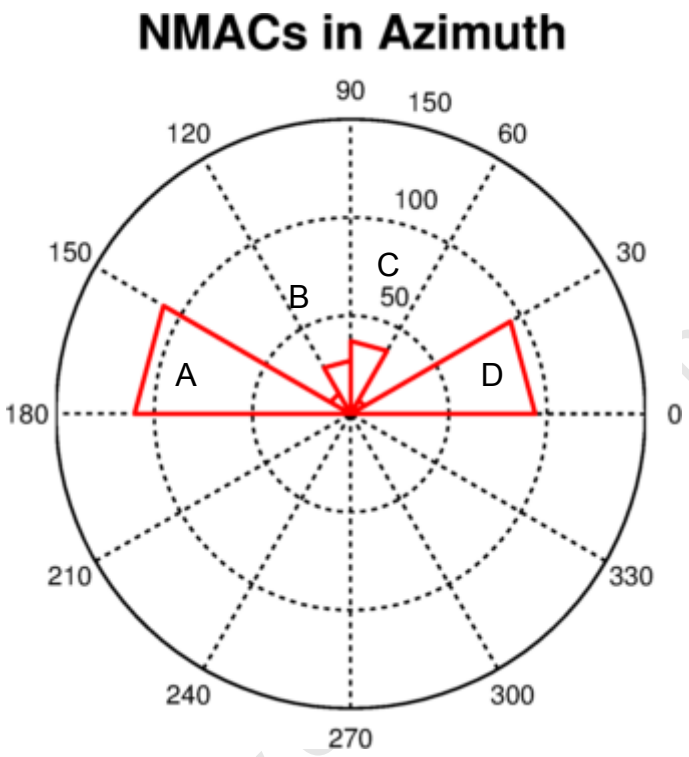

Figure 80: Azimuth angular histogram for NMAC data. The bins indicating collisions are shown in red, and labelled A to D.

The plot in Figure 80demonstrates that the majority of NMACs occur at azimuths of 0°-30° and 150° to 180°, which correspond to bins A and D. Bins B and C are smaller as less impacts occurred from the head-on direction.

A rose plot was created for the elevation data, using all TTIs (except for the 6s dataset). This plot is shown in Figure 81.

[12] A Rose Plot is an angular histogram [68].

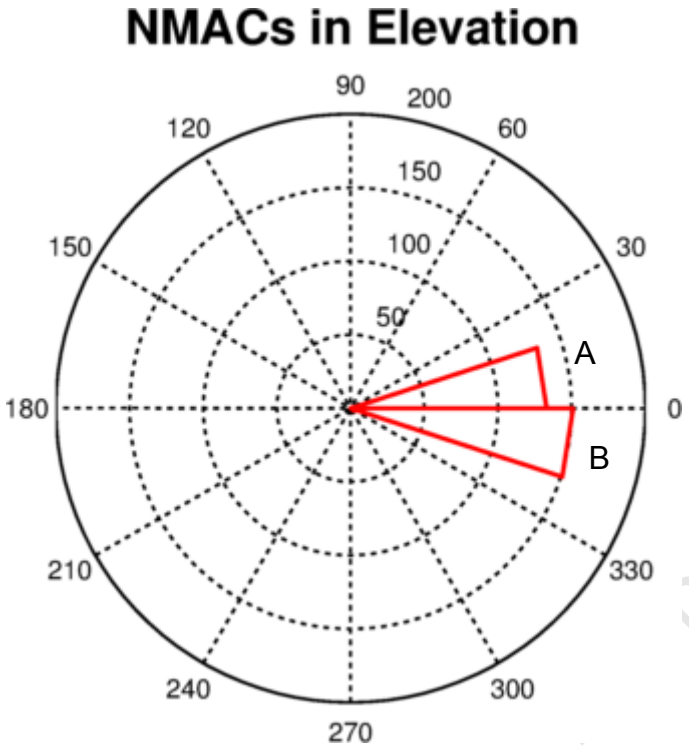

Figure 81: Elevation angular histogram for NMAC data. The collision bins where collisions occurred are labelled A and B.

In Figure 81 two clear collision bins are shown, both of these occurred in the elevation range -20° to 20°. As the figure shows, bin A had less than 130 collisions, whereas bin B had 150 collisions – i.e. more collisions coming from a lower altitude.

7.4.3 Time to Safety Data

The Time to Safety (TTS) was obtained by observing the Collision Cone trigger and is defined as the time taken to move the velocity vector out of the Collision Cone. However, it was found during preliminary simulations that this flag often triggered more than once during the encounter. This was attributed to the Nearly-constant velocity model implemented in Chapter 4, where sometime the threat would move back into the Collision Cone. Thus, it was decided for the sake of conservatism, that the *last* flag triggered would be used as the TTS.

A Box plot of TTS for the various TTIs is illustrated by Figure 82. This plot demonstrates that as the TTI decreases, the TTS increases. Also of particular importance is that the TTS is generally positively skew. In fact, it is shown that the most skew data set is TTI 25.

The reader will also notice various outliers in the data. Interestingly, there are outliers for TTI's 15 to 8 which are zero.

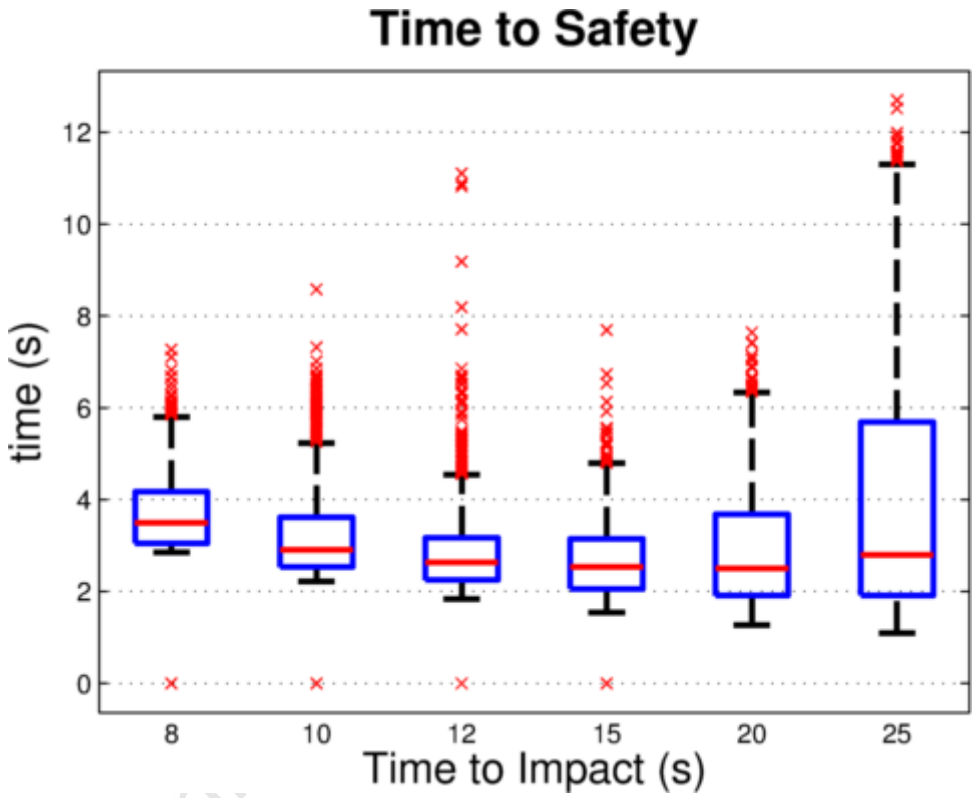

Figure 82: Box plot of the TTS data. The median increases as TTI decreases.

7.4.4 Summary

In summation, the data obtained from the Monte Carlo simulations illustrate that collision can be effectively evaded with a probability 78.6% at TTI 8s. For TTI's greater than 8s, this probability improves dramatically. Also, for a TTI of 6s, the UAV always experienced a NMAC.

The data also shows that most of the NMACs that did occur were at azimuth angles to the sides of the UAV. This will be discussed further in the proceeding chapter.

The Time to Safety (TTS) data indicated a positive skewness. In addition, it is inferred that TTS increases with a decrease in TTI. A number of outliers on both sides of the median were also noticed.

These findings will now be discussed analytically in the following chapter.

Chapter 8: Comparison & Discussion

In this chapter a discussion of the results obtained is provided. In addition, to this a comparison between CASSAM V.1 and CASSAM V.2 is also performed.

8.1 Comparison between CASSAM V.1 and CASSAM V.2

Essentially, both CASSAM V.1 and V.2 satisfy the same functional requirements discussed in Chapter 1. For the sake of completeness these are repeated and are:

- Evaluate
- Prioritise
- Declare
- Determine
- Command
- Execute

However, the manner in which they achieve these is where the discrepancy arises. Architecturally, the two systems are the same in that they both possess a collision detection section and a collision avoidance section. These differences are depicted below in Table 11.

Table 11: Comparison between CASSAM V.1 and CASSAM V.2.

System	CASSAM V.1	CASSAM V.2
Collision Cone	Collision Cone Algorithm	PCCA
SAM Controllers	NSAVDC, SATA	NSAVDC (with Clipping), SATA
Guidance Controller	Linear 3DVGC	Nonlinear 3DVGC

The Collision Cone algorithm employed CASSAM V.1 was essentially that of Watanabe [15]. This algorithm determined the Collision Cone on a plane spanned by and , allowing for 3D collision avoidance.

The Projected Collision Cone Algorithm (PCCA) projected the and vectors into the North-East plane. This enabled purely 2D manoeuvres where the aircraft only banked. In addition to this, the PCCA also implemented a Collision check Algorithm (CCA) to prevent false positives. Furthermore, the PCCA further extended the Collision Cone Algorithm of Watanabe [15] to be capable of handling threats approaching from behind the UAV.

The SAM controllers are largely the same between CASSAM V.1 and CASSAM V.2. The only difference is that the bank angle is clipped by the NSAVDC. This was done to ensure that the UAV does not experience large bank angles which could induce a stall as discussed in Section 2.5.

Both CASSAM V.1 and CASSAM V.2 employed a novel 3DVGC to guide the UAV out of the Collision Cone. The 3DVGC for both, consisted of a Direction Vector Algorithm (DVA) and a control algorithm. For CASSAM V.1, the control algorithm was linear and was designed on the assumption that the error angle dynamics were linear. In Chapter 5, it was proven that the error angle dynamics are in fact nonlinear and subsequently, a Nonlinear 3DVGC was implemented in CASSAM V.2. Phase Plane analysis also illustrated that the Nonlinear 3DVGC greatly outperformed the Linear 3DVGC of CASSAM V.1. Furthermore, both systems employed the same DVA.

The simulation results for CASSAM V.1 and CASSAM V.2 demonstrate that both evade collision successfully. However, the manner in which the respective systems manoeuvre needs to be discussed.

In all test scenarios, CASSAM V.1 commanded the UAV to dive. This can be attributed to the way in which the Collision Cone is calculated. As stated previously, the Collision Cone algorithm of CASSAM V.1 calculates the cone by using the plane spanned by and . Thus, even if the UAV and threat are flying in the North East plane, the cone can be formed by a perpendicular plane. This arises when either the Threat or the UAV velocity vectors contain a component in the Down axis. This is illustrated by Figure 83.

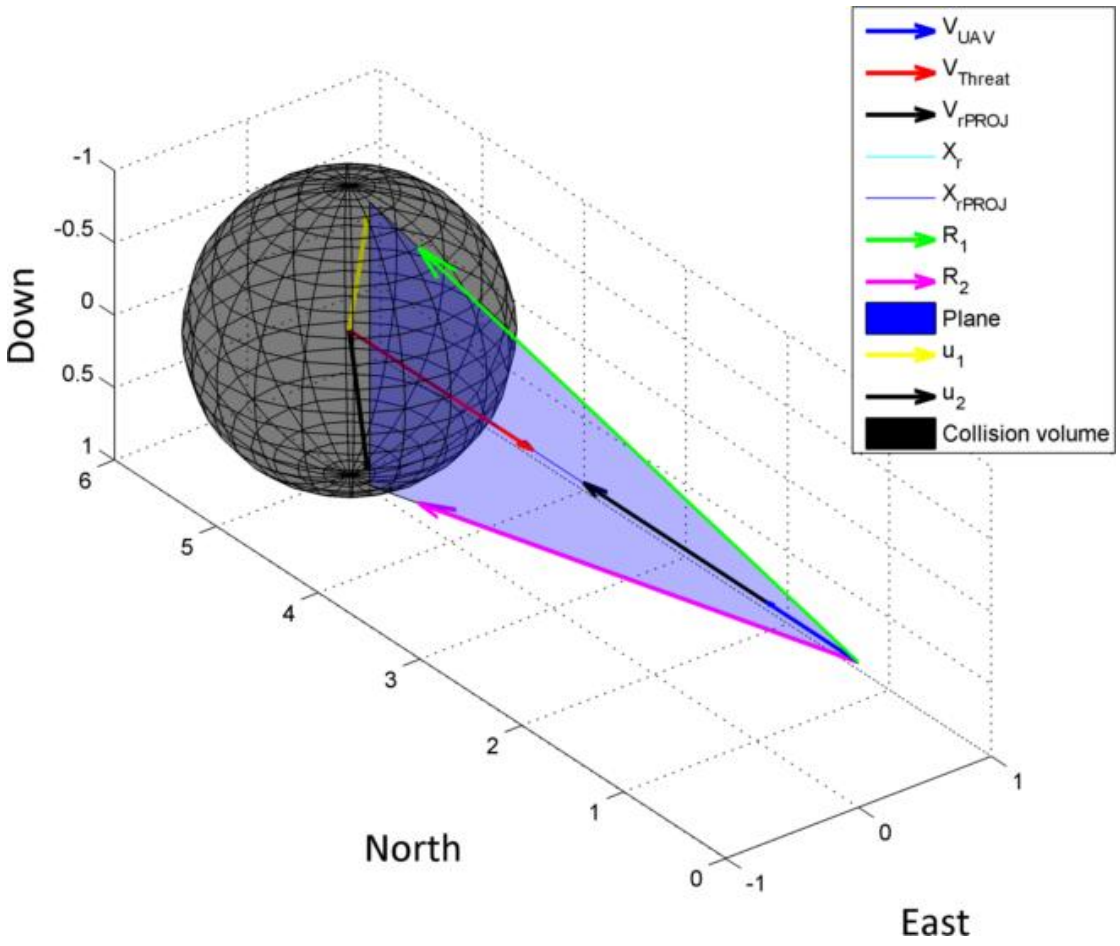

Figure 83: Collision Cone of CASSAM V.1. The reader should note that even though the UAV and Threat are flying in the North East plane; the algorithm calculates the cone plane to be perpendicular to it.

Thus, considering that the Nearly Constant Velocity model implemented in Chapter 4, there will always be a velocity component in the Down axis (albeit a minuscule one). This was due to the acceleration noise.

Comparatively, the PCCA employed by CASSAM V.2, always calculates the cone to be in the North-East plane. This is due to it projecting and into this plane. This produces reference commands to the 3DVGC which always lie in the North-East plane.

The dive manoeuvres commanded by CASSAM V.1 would result in an increase in airspeed. This was not perceived in this simulation as a constant velocity model was implemented (See Section 4.3). But in reality, if the UAV is in a dive, the airspeed would increase as the UAV would be accelerating by the force of gravity. Now, with reference to Section 2.5, all aircraft have a Never Exceed Speed or which, when exceeded, will cause damage to the aircraft. Hence, collision avoidance manoeuvres commanded by CASSAM V.1 would result in damage to the aircraft.

Furthermore, diving rapidly could also result in collision with the ground. This can occur if the collision avoidance system is activated close to the ground. Moreover, diving could also result in collision with other oncoming traffic flying on flight levels below the UAV.

Considering the aforementioned consequences of diving, it is clear that CASSAM V.2 is the superior system both in terms of safety as well as performance.

The reader should note that as was illustrated in Section 6.5.4, CASSAM V.2 does cause the UAV to pitch up slightly. This arises because the underlying NSA virtual actuator will pitch up to generate more normal specific acceleration. However, in a practical application after collision is successfully avoided the UAV's autopilot would re-engage. The autopilot would then regulate the pitch angle back to an acceptable value and continue the mission. Thus, this behaviour is deemed as adequate.

8.2 Discussion of Monte Carlo Data

The data produced by the Monte Carlo simulations provided significant insight into the robustness of CASSAM V.2. The probability of Near Mid-Air Collision was particularly enlightening. From Figure 78 in Chapter 7, it can be seen that that P(NMAC) is inversely proportional to TTI. This is intuitive, as when the TTI decreases, the UAV has less time to react and execute an avoidance manoeuvre. Thus, as the TTI decreases, a NMAC is more likely to occur. This has an impact on the selection of an appropriate sensing device and also on the processing required for a sense and avoid system.

So, to successfully avoid collisions using CASSAM V.2, a minimum detection requirement of at least 15s TTI is required. For a head on collision with a threat aircraft flying at 150 m/s and a UAV cruising at 35 m/s, this relates to a distance of 2.8 km. This distance is more than feasible for a modern radar system.

The reader should however note that the simulation did assume that the threat was moving at a nearly constant velocity. The effects of a manoeuvring threat will most certainly provide a more precise P(NMAC) distribution.

The spatial distribution plots also revealed that the majority of NMACs occurred in azimuths to the left and right of the UAV. This is illustrated appropriately by Figure 80. The result is attributed to the fact that in those scenarios to effectively guide the UAV out of the Collision Cone requires more control effort. This is exacerbated by the UAV being constrained to avoid collisions using one degree of freedom (turning) only. Essentially, during these scenarios the UAV physically cannot manoeuvre any faster out of the cone due to the physical limitations of the aircraft. This is illustrated below by Figure 84 where it is shown that the UAV physically cannot turn and accelerate any further and thus, collision occurs.

Thus, this places a further constraint on the selection of the sensor selected for use by the sense and avoid system. The sensor must have a wide field of regard (FOR). In fact, for effective collision avoidance using CASSAM V.2 the azimuths of ± 90° should have detection of at least 15s TTI or better.

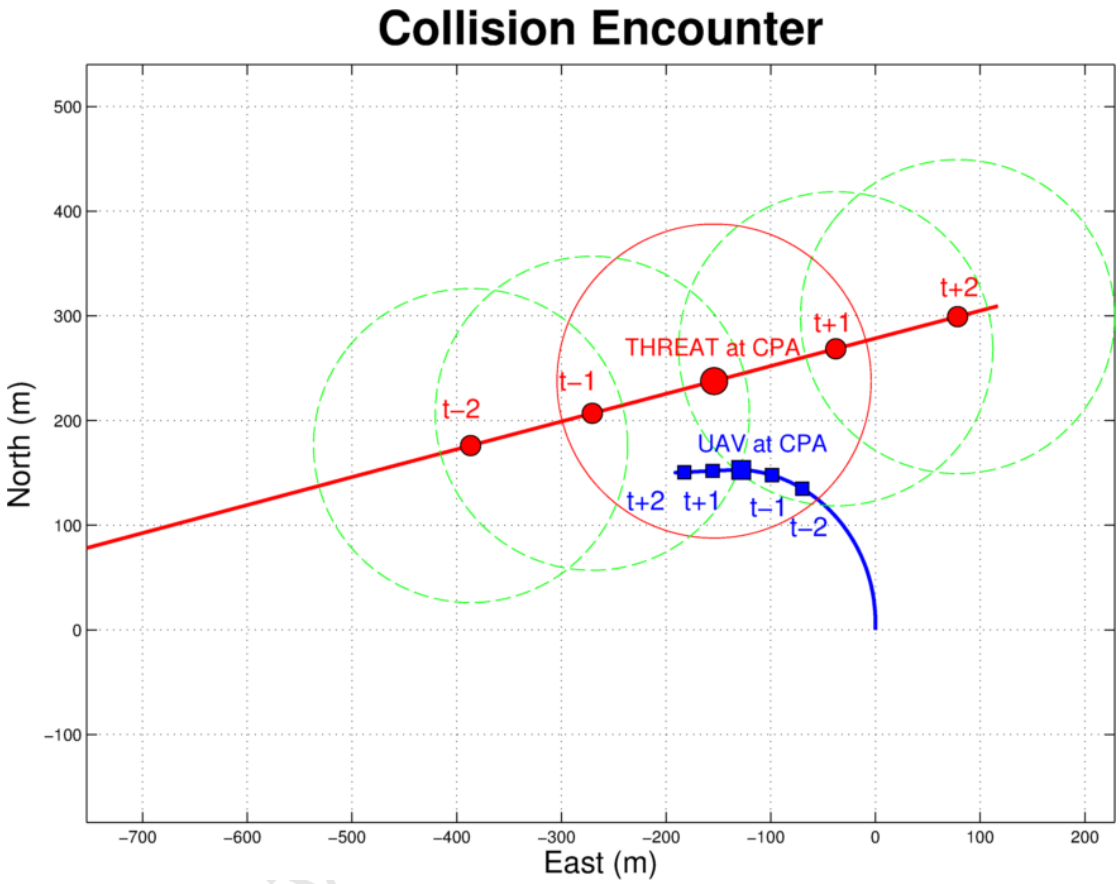

Figure 84: 2D plot of a Collision Encounter where a NMAC has occurred. The threat is initially situated at a bearing -90° from the UAV. 't' represents the time of CPA.

The elevation NMAC data was inconclusive as shown by Figure 81. There is not much difference in the number of NMACs in the two elevation bins. Thus, it is difficult to determine whether or not ascending or descending threats pose a greater risk.

The Time-to-Safety (TTS) data was plotted in Figure 82. Firstly, it can be seen from the data that the median of TTS is inversely proportional to TTI. Once

more, this is expected behaviour because when the TTI decreases the collision cone grows larger more quickly. Thus, with a larger collision cone, it takes more time to guide the velocity vector out of the cone.

Also noticed from the data are a significant number of outliers. These are attributed to the nearly constant velocity threat motion. During the scenario once collision has been avoided, the threat moves back into the cone. As discussed in Chapter 7, this results in the Collision Cone flag being pulled high again. Future applications of CASSAM V.2 should include a form of hysteresis to circumvent this occurring.

There also exist a certain number of outliers for TTI 15, 12, 10 and 8 which are zero. This is a result of the fact that in plotting the Figure 82, the scenarios where NMACs occurred were also included. When an NMAC happens, the Collision Cone flag remains high, thus no falling edge is produced. This implies that no TTS exist, because “safety” was never attained. But Matlab initiates the “tts” variable to zero and this is what is seen in the plot.

TTI 25 and TTI 20 also demonstrate significant positive skewness for the TTS variable. This can once more be attributed to the target motion causing the Collision Cone to re-engage. This occurs more so than in the other TTIs as the time to impact is larger, providing more time for the threat to weave in and out of the Collision Cone. This caused larger TTS values to be logged by the Monte Carlo script.

8.3 Summary

This chapter provided a discussion of the data produced by the deterministic test scenarios as well as the Monte Carlo simulations. A comparison was also made between CASSAM V.1 and CASSAM V.2 where it was shown that

CASSAM V.2 was the superior system. The main points are summarised below:

- Comparisons between CASSAM V.1 and CASSAM V.2 showed that V.1 made the UAV dive rapidly to avoid an oncoming threat, but V.2 took the approach of going around the threat.
- CASSAM V.1 had the drawback of potentially wasting fuel and increasing the risk of damaging the UAV in comparison to CASSAM V.2 that took the less costly and risky approach.
- The Monte Carlo simulation confirmed the robustness of CASSAM V.2, through the use of statistical analysis.

Chapter 9: Conclusion

This chapter concludes the thesis with a summary of the work presented and highlights the significance of the research towards the field. Recommendations for future research on this particular topic are made and improvements are also proposed.

9.1 Conclusions of Research

Two collision avoidance systems were developed based on the notion of Specific Acceleration Matching (SAM) Control. The first system was called CASSAM V.1 and the second CASSAM V.2. Both of these were successfully implemented and simulated in specific test scenarios.

These respective systems were the product of an extensive thematic survey of the literature in the field. Geometric methods were chosen for Collision Detection for their inherent simplicity. It was also decided that an acceleration based method would be employed as this was proven in the literature. However, the acceleration methods lacked an interface to the UAV as well as commanding lateral acceleration which produced sideslip. SAM Control addressed these shortcomings and was integrated into the respective collision avoidance systems.

A variation of the original Collision Cone algorithm was implemented in CASSAM V.2. This algorithm extended the conventional Collision Cone Algorithm to produce a new algorithm, the Projected Collision Cone Algorithm (PCCA), which addressed the shortcomings of CASSAM V.1. The PCCA only produced commanded manoeuvres which steered the aircraft laterally only to avoid collisions,

Furthermore, a novel control algorithm the Linear 3DVGC was developed for CASSAM V.1. This controller was designed under the assumption of linear

dynamics. However, it was proven during the design of CASSAM V.2 that this assumption was incorrect and the dynamics were in fact nonlinear. A new controller, the Nonlinear 3DVGC was then implemented for CASSAM V.2. Phase Plane Analysis demonstrated that this controller produced superior behaviour to the Linear 3DVGC.

During the testing of CASSAM V.1 it was discovered the system commanded the UAV to dive rapidly to avoid collision. This was regularly shown in simulations results, for example in Figure 45 and Figure 47. This was undesirable and dangerous behaviour. It was discussed in Section 8.1 (and illustrated by Figure 83) that the manoeuvres were actually caused by the manner in which CASSAM V.1 calculated the Collision Cone.

Comparatively, CASSAM V.2 remedied this and only commanded the UAV to turn to avoid collisions as seen by Figure 68 and Figure 70. Thus, considering the safety aspects as well as performance, CASSAM V.2 was deemed the superior system.

Finally, CASSAM V.2 was tested by use of Monte Carlo Simulations. These simulations produced statistical data as is evident Figure 78. This data was broadly supportive of the use of CASSAM V.2 as a means to answer the sense and avoid dilemma for UAVs.

9.2 Significance of Research

During the course of this research project, the following significant contributions were made to the field:

- This research resulted in the first known application of SAM Control to the UAV collision avoidance problem. From the results produced, it is indeed a viable solution as it is computationally efficient as well as

robust. SAM Control for collision avoidance also draws on the shortcomings on the common Proportional Navigation methods employed in the majority of literature.

- The conventional Collision Cone algorithm was extended to the Projected Collision Cone Algorithm (PCCA). This was done to enable avoidance of threats moving in 3D by use of 2D evasive manoeuvres. Furthermore, the PCCA also provided a means to evade threats approaching the UAV from behind.
- A nonlinear guidance law, the Nonlinear 3DVGC, was also developed to effectively manoeuvre the UAV to safety. This was achieved by means of acceleration commands to the UAV which would not command an increase or decrease in airspeed. Phase Plane Analysis showed the viability of this controller under different initial conditions.
- A simulation environment was developed to test the collision avoidance system. This simulation environment included a Nearly Constant Velocity model for the threat as well as a kinematic model for the UAV. This environment will prove invaluable for future research and can be used as a test bed for verification of new collision avoidance algorithms.
- A Monte Carlo simulation script was also developed in this research. This Matlab script extended the Collision Encounter simulation by providing it with a batch capability. Additionally, a Random Threat Generation algorithm was created, to allow random encounter scenarios to be created. Also, a Collision Monitor algorithm was developed to monitor the simulation accordingly.

9.3 Recommendations for Future Research

A natural direction for future research is extending CASSAM V.2 to evade multiple threats. This is a non-trivial task as the implementation of the PCCA will need to be considered carefully. One solution could be to calculate an individual cone for each threat, then prioritise the largest cone. The UAV will then be steered out of that cone. Another solution could be to take each of the

individual cones and sum them together to form one cone. This large cone will encapsulate all possible collisions. The guidance algorithm would then need to steer the UAV out of this cone.

The reader will notice that the PCCA in CASSAM V.2 essentially provides heading commands to the 3DVGC. It is the Author's view that an investigation into utilizing a conventional heading controller for collision avoidance should be performed. Most UAV autopilots possess a heading controller as part of their suite of control loops. Thus, if a collision avoidance system can be demonstrated with a conventional heading controller, the system will be extendable to a vast number of UAV platforms.

Operational aspects should also be researched. In a practical UAV system, the Ground Control operator needs to be made aware of the status of the UAV and decide if action is necessary. One manner to achieve this safely is to allow the collision avoidance system to send a message to the operator that a collision is imminent. If the operator does not respond in a timely manner, the collision avoidance system will then steer the UAV to safety.

The encounter simulation should be improved by modelling a more realistic threat model. This model should include the effects of manoeuvring and can be implemented by a Singer Model [59]. Furthermore, multiple threats should also be modelled and the encounter simulation should be extended accordingly.

An improved UAV model could also be used for the encounter simulations. The current simulation employs a kinematic model but to factor the aerodynamic and engine effects a full nonlinear kinetic model should be implemented. Furthermore, the UAV autopilot should also be modelled or be incorporated to form a Hardware in the Loop Simulation (HILS).

Lastly, CASSAM V.2 should be integrated with threat detection and tracking/estimation algorithms. Currently, CASSAM V.2 is provided perfect states from the simulation but a more realistic simulation would be to include estimation and tracking algorithms in the loop. The integrated system will then form part of a greater sense and avoid system which can then be tested in simulation before integration on to hardware.

9.4 Summary

Specific Acceleration Matching Control and Collision Cone algorithms were examined and further developed for application to the collision avoidance problem for UAVs. A simulation environment was developed where pre-defined test scenarios were tested. Monte Carlo simulations were also performed to validate the robustness of the collision avoidance system. The results of the aforementioned were discussed and analysed in detail and concise conclusions were drawn.

Appendix A: Additional Mathematics

This appendix represents the additional mathematics employed in this thesis. It will serve as a reference for the algorithms described in the thesis itself. The purpose of this chapter is focused on the application of these equations and not on the derivation.

A.1 Attitude Representation

Various methods exist to describe the orientation of an axis system with respect to another. In this section the methods utilized in this thesis will be described.

A.1.1 Direction Cosine Matrix

The Direction Cosine Matrix [67] is a 3 x 3 matrix that describes the attitude of an axis system with reference to another axis system. The matrix describes the attitude of axis system B with respect to axis system A. The rows of are defined by the unit vectors of the axis system B. coordinated in A. Thus, to transform a vector from axis A to B:

(i)

Diebel [67] also states that because the DCM is an orthogonal matrix, its inverse is in fact its transpose. Thus, the following relationship holds true:

(ii)

The Direction Cosine Matrix provides a very complete solution to attitude representation. However, considering it requires nine elements and also that it is not very intuitive, it is often converted to Euler Angles.

The conversion to Euler (3-2-1) Angles is provided by [67] and is shown to be:

——— (iii)

(iv)

——— (v)

A.1.2 Axis-Angle Representation

The Axis-Angle [61] representation describes a rotation as *a unit vector and an angle*. This is illustrated graphically by Figure 85.

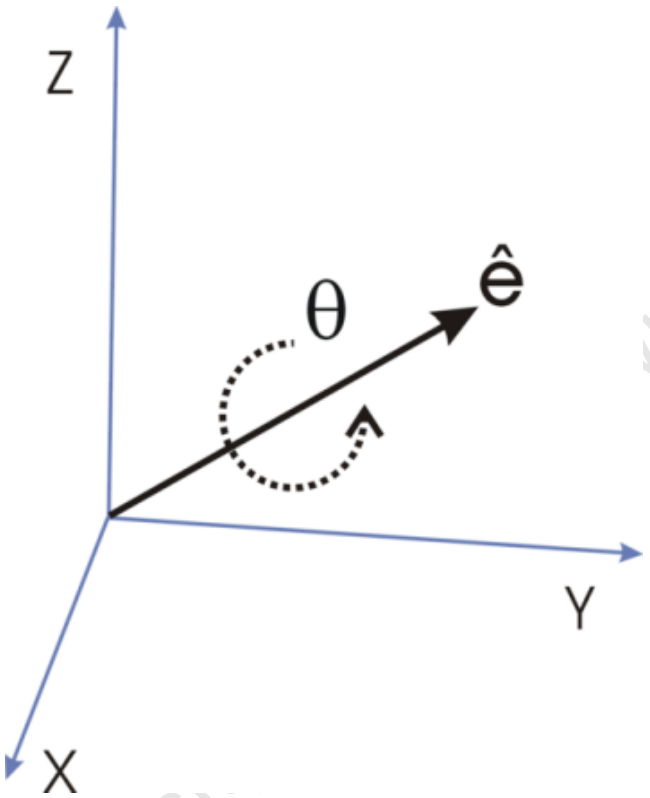

Figure 85: The Axis-Angle method represents a rotation by use of a unit vector and an angle [68].

This is convenient when one has two vectors and desires to know the rotational relationship between the two. This is achieved by taking the cross product of the two vectors then normalising this vector. This becomes the "Axis". Then, to determine the angle one can employ the dot product [61].

This Axis-Angle representation can be converted to a rotation matrix by the following equation provided by Baker [61]:

(vi)

Where, and are the cosine and sine of the rotation angle respectively. is defined by:

(vii)

Also, , and are the components of the axis vector.

A.2 Cylindrical Coordinate System

The Cylindrical Coordinate System [69] is a 3D coordinate system representing particle motion. The position ***P*** defined in Cylindrical Coordinates is illustrated below in Figure 86.

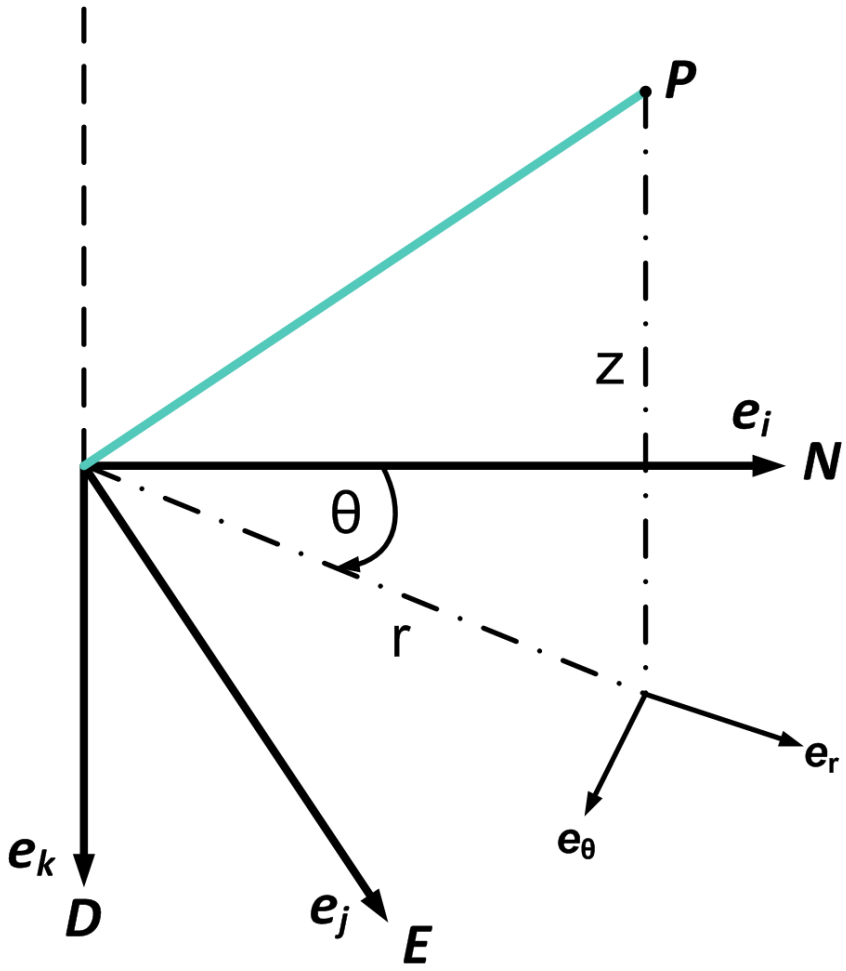

Figure 86: Cylindrical Coordinate System vector diagram. The reader should note that the NED Frame has been used as the Rectangular frame of reference.

The position of ***P*** is defined by r,θ,z. In terms of the NED frame, these become:

(viii)

– (ix)

(x)

The velocity vector in Cylindrical Coordinates is given by [69] and is shown to be,

(xi)

where are the unit vectors along which the velocity components are defined and are shown in Figure 86. The scalars are obtained from the following equations:

(xii)

(xiii)

(xiv)

A.3 Spherical Coordinate System

The Spherical Coordinate System [69] is a 3d coordinate system describing the motion of particle using radial distance (ρ) and two angles ( ). The position ***P*** defined in Spherical coordinates is illustrated below in Figure 87.

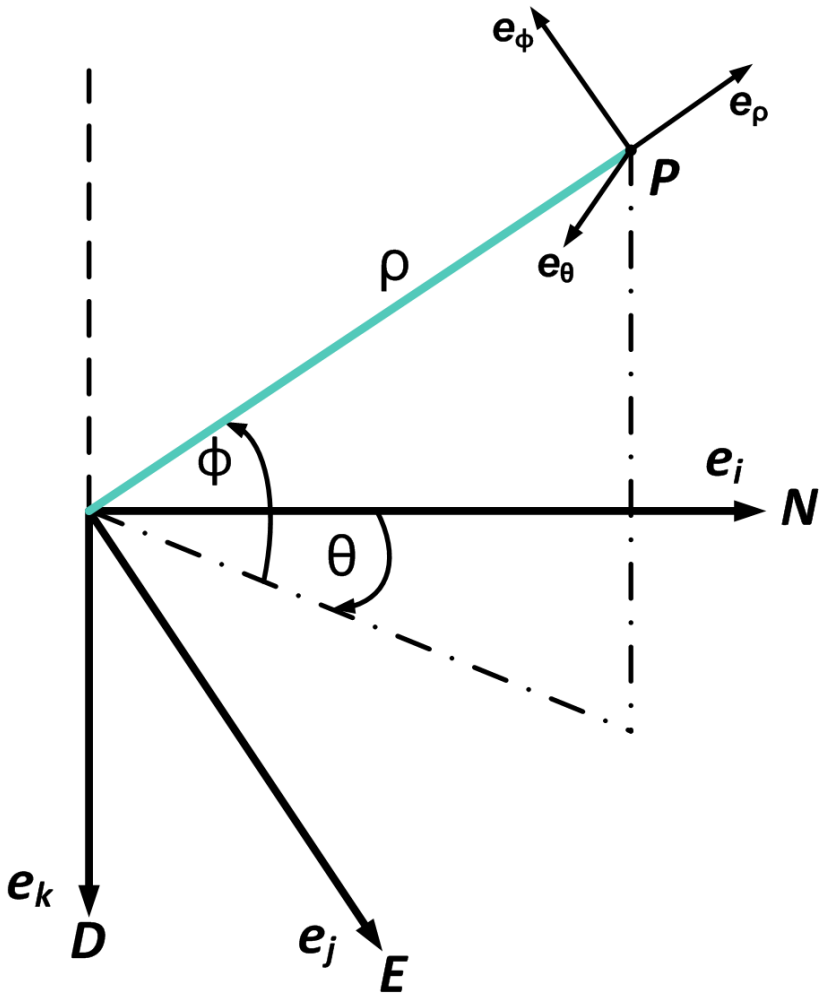

Figure 87: Spherical Coordinate System vector diagram.

The position of ***P*** is defined by . In terms of the NED frame, these become:

(xv)

– (xvi)

– (xvii)

The velocity vector in Spherical Coordinates is given by [69] and is shown to be,

(xviii)

where are the unit vectors along which the velocity components are defined and are shown in Figure 87. The scalars are obtained from the following equations:

(xix)

(xx)

(xxi)

–

Bibliography

[1] Department of Defence - United States of America, "FY2009-2034 Unmanned Systems Integrated Roadmap," 2009.

[2] Jeff Wise. (2009, October) Popular Mechanics. [Online]. http://www.popularmechanics.com/science/space/4213464

[3] Lacher, Maroney, and Zeitlin, "Unmanned Aircraft Collision Avoidance - Technology Assesment and Evaluation Methods," The MITRE Corporation, 2007.

[4] Federal Aviation Administration (FAA), "Sense and Avoid (SAA) for Unmanned Aircraft Systems (Final Report)," 2009.

[5] Federal Aviation Administration. (2010, October) FAA - TCAS Definition. [Online]. http://adsb.tc.faa.gov/TCAS.htm

[6] NATO Naval Armaments Group Joint Capability Group on Unmanned Aerial Vehicles , "Sense and Avoid Requirements for Unmanned Aerial Vehicle Systems Operating in Non-Segregated Airspace," NATO, Requirements Standard PFP(NNAG-JCGUAV)D(2008)0002, 2008.

[7] FAA. (2008, May) FAA - Midair Collision Avoidance. [Online]. http://www.faa.gov/about/office_org/headquarters_offices/ato/tracon/anchorage/pilots_info/mca/

[8] J. Kuchar and L. Yang, "Survey of Conflict Detection and Resolution Modeling Methods," in *AIAA Guidance, Navigation, and Control Conference*, New Orleans, LA, 1997.

[9] J. Kuchar and L. Yang, "A Review of Conflict Detection and Resolution Modeling Methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179-189, December 2000.

[10] A. Mujumdar and R. Padhi, "Nonlinear Geometric and Differential Geometric Guidance of UAVs for Reactive Collision Avoidance.," Indian Institute of Science - Department of Aerospace Engineering , Bangalore,

India, Technical Report 2009.

[11] Chakravarthy and Ghose, "Obstacle Avoidance in a Dynamic Environment: A Collision Cone Approach," *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans*, vol. 28, no. 5, 1998.

[12] L. Tychonievich et al., "A Maneuvering-Board Approach to Path Planning with Moving Obstacles.," in *International Joint conference on Artificial Intelligence (IJCAI)*, 1989, pp. 1017-1021.

[13] P. Fiorini and Z. Schiller, "Motion planning in dynamic environments using the relative velocity paradigm," in *IEEE Conference on Robotics and Automation*, 1993, pp. 560-565.

[14] B. Damas and J. Santos-Victor, "Avoiding Moving Obstacles: The Forbidden Velocity Map," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 4393-4398.

[15] Y Watanabe, "Stochastically Optimized Monocular Vison-Based Guidance," Georgia Institute of Technology, PhD Thesis 2008.

[16] A Melander, "Quadrotor Implementation of the Distributed Reactive Collision Avoidance Algorithm," University of Washington, MSc Thesis 2010.

[17] B. Kluge and E Prassler, "Reflective Navigation: Individual Behaviours and Group Behaviours," in *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, New Orleans, LA, 2004, pp. 4172-4177.

[18] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic Obstacle Avoidance in uncertain environment combining PVOs and Occupancy Grid," in *IEEE Conference on Robotics and Automation*, Roma, Italy, 2007, pp. 1610-1666.

[19] K.Y. Kim, J.W. Park, and M.J. Tahk, "UAV Collision Avoidance Using Probabilistic Method in 3D," in *International Conference on Control,*

Automation and Systems, Seoul, Korea, 2007, pp. 826-829.

[20] F. Lindsten and P.J, Gustafsson, F. Nordlund, "Conflict Detection Metrics for Aircraft Sense and Avoid Systems," in *Proceedings of the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processe*, Barcelona, Spain, 2009.

[21] S.Y. Ghangrekar, "A Path Planning and Obstacle Avoidance Algorithm for an Autonomous Robotic Vehicle," University of North Carolina at Charlotte, Charlotte, MSc Thesis 2009.

[22] P. Lester. (2005, July) A* Pathfinding for Beginnners. [Online]. http://www.policyalmanac.org/games/aStarTutorial.htm

[23] S.M. LaValle, *Planning Algorithms*.: Cambridge University Press, 2006.

[24] F Belkhhoch, "Reactive Path Planning ina Dynamic Environment," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 902-911, August 2009.

[25] LaValle S.M., "Rapidly-Exploring Random Trees: A new tool for path planning," Iowa State University - Computer Science Department, Report 1998.

[26] Griffiths S. et al., "Maximizing Miniature Aerial Vehicles," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 34-43, 2006.

[27] C. Fulgenzi and A. Laugier, C. Spalanzani, "Probabilistic Rapidly-exploring Random Trees for autonomous navigation among moving pedestrians," in *IEEE International Conference on Robotics and Automation*, Kobe, Japa, 2009.

[28] L. Chang-an, C. Jin-gang, L. Guo-dong, and L. Chun-yang, "Mobil Robot Path Planning Based on an Improved Rapid-exploring Random Tree in Unknown Environment," in *Proceedings of the IEEE International Conference on Automation and Logistics*, Qingdao, China, 2008, pp. 2375-2379.

[29] J. Kuffner. (2009) Classic Path Planning with RRTs. [Online].

http://www.kuffner.org/james/plan/

[30] Han and Bang, "Proportional Navigation-Based Optimal Collision Avoidance for UAVs," in *2nd International Conference on Autonomous Robots and Agents*, Palmerston North, New Zealand, 2004.

[31] S Han, "Proportional Navigation-Based Optimal Collision Avoidance for UAVs," Korea Advanced Institute of Science and Technology, Thesis 2005.

[32] Han, Bang, and Yoo, "Proportional Navigation-Based Collision Avoidance for UAVs," *International Journal of Control, Automation and Systems*, vol. 7, no. 4, pp. 553-565, 2009.

[33] George and Ghose, "A Reactive Inverse PN Algorithm for Collision Avoidance among Multiple Unmanned Vehicles," in *American Control Conference* , St. Louis, MO, USA, 2009, pp. 3890-3895.

[34] A.L. Smith, "UAS Collision Avoidance Algorithm that Minmizes the Impact on Route Surveillance," Air Force Institute Technology, Wright-Patterson Air Force Base, Ohio, USA, MSc Thesis 2009.

[35] G.M. Siouris, *Missile Guidance and Control Systems*. USA: Springer, 2004.

[36] D.H. Shim and S. Sastry, "An Evasive Maneuvering Algorithm for UAVs in See-and-Avoid Situations," in *American Control Conference*, New York City, USA, 2007, pp. 3886-3891.

[37] J. Goss, R. Rajvanshi, and K. Subbaro, "Aircraft conflict Detection and Resolution using Mixed Geometric and Collision Cone Approaches," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Providence, RI, USA, 2004.

[38] I. K. Peddle, "Acceleration Based Manoeuvre Flight Control System for Unmanned Aerial Vehicles," Stellenbosch University, Cape Town, PhD Thesis 2008.

[39] D.R. Gaum, "Aggresive Flight Control for a Fixed-Wing Unmanned Aerial Vehicle," Stellenbosch University, Cape Town, Masters Thesis 2009.

[40] D. Blaauw, "Flight Control System for a Variable Stability Blended-Wing-Body Unmanned Aerial Vehicle," Stellenbosch University, Cape Town, Masters Thesis 2009.

[41] R. D. de Hart, "Advanced Take-off and Flight control Algorithms for Unmanned Aerial Vehicles," Stellenbosch University, Cape Town, Masters Thesis 2009.

[42] L.D. Reid B. Etkin, *Dynamics of Flight, Stability and Control*.: Wiley & Sons, 1996.

[43] NASA. (2011) NASA History Page. [Online]. http://history.nasa.gov/SP-367/appendc.htm#f166

[44] M.V. Cook, Flight Dynamics Priniciples: A Linear Systems Approach to Aircraft Stability and Control, 2nd ed.: Butterworth-Heinemann, 2007.

[45] The Pilot Ground School. (2006) Load factors: airplane operating limits. [Online]. http://www.free-online-private-pilot-ground-school.com/Load_factors.html

[46] M. Cavcar. (2004) Anadolu University - School of Civil Aviation. [Online]. http://home.anadolu.edu.tr/~mcavcar/common/Loadfactor.pdf

[47] D.P. Raymer, *Aircraft Design: A Conceptual Approach*, 4th ed., J. A. Schetz, Ed. Blacksburg, USA, Virginia: AIAA, 2006.

[48] M.M. Basson, "Stall Prevention Control for Fixed Wing Unmanned Aerial Vehicles," Stellenbosch University, Stellenbosch, MSc Thesis 2010.

[49] L.J. Clancy, *Aerodynamics*. London: Pitman Publishing Limited, 1975.

[50] S. Williams. (2011) Aviation Glossary. [Online]. http://aviationglossary.com/aircraft-terms-definition/v-g-diagram/

[51] FAA, Pilot's Handbook of Aeronautical Knowledge., 2010.

[52] J. Lin. (2011, May) EE Times - Measuring Return on Investment of Model Based Design. [Online]. http://www.eetimes.com/design/military-aerospace-design/4216303/Measuring-return-on-investment-of-model-based-design

[53] M Anthony, M Behr, M Jardin, and R. Ruff, "Model-Based Design for Large High-Integrity Systems: A Discussion on Verification and Validation," The Mathworks, White Paper 2010.

[54] J. Tung. (2007, September) EE Times - Using Model-Based Design to Test and Verify Automotive Embedded Software. [Online]. http://www.eetimes.com/design/eda-design/4018506/Using-Model-Based-Design-to-Test-and-Verify-Automotive-Embedded-Software

[55] A Turevskit, S Gage, and C. Buhr, "Model-Based Design of a New Light-Weight Aircraft," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Hilton Head, South Carolina, USA, 2007.

[56] A Wakefield and S. Miller, "Improving System Models Using Monte Carlo Techniques on Plant Models," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Honolulu, Hawaii, 2008.

[57] AOPA Air Safet Foundation. (2009, June) ASF - Airspace for Everyone. [Online]. http://www.aopa.org/asf/publications/sa02.pdf

[58] M.J. Kochenderfer, L.P Espindle, J.K. Kuchar, and J.D Griffith, "A Comprehensive Aircraft Encounter Model of the National Airspace System," *Lincoln Laboratory Journal*, vol. 17, pp. 41-53, February 2008.

[59] X.R. Li and V.P. Jilkov, "Survey of Maneuvering Target Tracking. Part 1: Dynamic Models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333-1364, October 2003.

[60] A. Gupta and R. Padhi, "Nonlinear Geometric and Differential Geometric Guidance of UAVs with Vision Sensing for Reactive Obstacle Avoidance," Indian Institute of Science - Department of Aerospace, Bangalore, India, Technical Report 2010.

[61] Martin Baker. (2010) euclideanspace - Axis Angle. [Online]. http://www.euclideanspace.com/maths/geometry/rotations/conversions/angleToMatrix/index.htm

[62] M Braae, *Control Engineering - 1*. Cape Town, South Africa: Martin Braae, 2001.

[63] Bob Kirkby. (2000) Skywalker - Understanding the Spiral Dive. [Online]. http://www.skywalker.ca/Flying_Stories/Bob_s_Stories/SpiralDive.PDF

[64] J.E. Slotine and W Li, *Applied Nonlinear Control*. NJ: Prentice Hall, 1991.

[65] P. Dawkins. (2011) Pauls Online Notes: Differential Equations. [Online]. http://tutorial.math.lamar.edu/Classes/DE/PhasePlane.aspx

[66] G.F. Franklin, J.D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Upper Saddle River, New Jersey, United States of America: Pearson Prentice Hall, 2009.

[67] J. Diebel, "Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors," Stanford University, Stanford, California, Report 2006.

[68] Wikipedia. (2011) Wikimedia Commons web page. [Online]. http://commons.wikimedia.org/wiki/Main_Page

[69] J.L. Meriam and L.G. Kraige, *Engineering Mechanics - Volume Two: Dynamics (SI Edition)*, 5th ed. United States of America: John Wiley & Sons, 2003.

[70] the MathWorks. (2011, January) MathWorks. [Online]. http://www.mathworks.com/help/techdoc/ref/rose.html