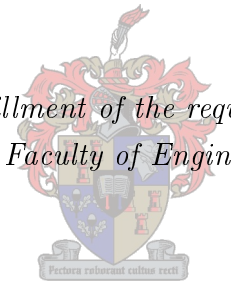
Robust Sampling-Based Conflict Resolution for Commercial Aircraft in Airport Environments

by
William van den Aardweg



Thesis presented in fulfillment of the requirements for the degree of Master of Engineering in the Faculty of Engineering at Stellenbosch University

Supervisors:
Mr JAA Engelbreght, Dr CE van Daalen

Department of Electrical and Electronic Engineering

March 2015

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

December 2014

Abstract

This thesis presents a robust, sampling-based path planning algorithm for commercial airliners that simultaneously performs collision avoidance both with intruder aircraft and terrain. The existing resolution systems implemented on commercial airliners are fast and reliable; however, they do possess certain limitations. This thesis aims to propose an algorithm that is capable of rectifying some of these limitations. The development and research required to derive this conflict resolution system is supplied in the document, including a detailed literature study explaining the selection of the final algorithm. The proposed algorithm applies an incremental sampling-based technique to determine a safe path quickly and reliably. The algorithm makes use of a local planning method to ensure that the paths proposed by the system are indeed flyable. Additional search optimisation techniques are implemented to reduce the computational complexity of the algorithm. As the number of samples increases, the algorithm strives towards an optimal solution; thereby deriving a safe, near-optimal path that avoids the predicted conflict region. The development and justification of the different methods used to adapt the basic algorithm for the application as a conflict resolution system are described in depth. The final system is simulated using a simplified aircraft model. The simulation results show that the proposed algorithm is able to successfully resolve various conflict scenarios, including the generic two aircraft scenario, terrain only scenario, a two aircraft with terrain scenario and a multiple aircraft and terrain scenario. The developed algorithm is tested in cluttered dynamic environments to ensure that it is capable of dealing with airport scenarios. A statistical analysis of the simulation results shows that the algorithm finds an initial resolution path quickly and reliably, while utilising all additional computation time to strive towards a near-optimal solution.

Opsomming

Hierdie tesis bied 'n robuuste, monster-gebaseerde roetebeplanningsalgoritme vir kommersiële vliegtuie aan, wat botsingvermyding met indringervliegtuie en met die terrein gelyktydig uitvoer. Die bestaande konflik-vermyding-stelsels wat op kommersiële vliegtuie geïmplementeer word, is vinnig en betroubaar; dit het egter ook sekere tekortkominge. Hierdie tesis is daarop gemik om 'n algoritme voor te stel wat in staat is om sommige van hierdie tekortkominge reg te stel. Die ontwikkeling en navorsing wat nodig was om hierdie konflik-vermyding-algoritme af te lei, word in die dokument voorgelê, insluitende 'n gedetailleerde literatuur-studie wat die keuse van die finale algoritme verduidelik. Die voorgestelde algoritme pas 'n inkrementele, monster-gebaseerde tegniek toe om vinnig en betroubaar 'n veilige roete te bepaal. Die algoritme maak gebruik van 'n lokale beplanningsmetode om te verseker dat die roetes wat die stelsel voorstel inderdaad uitvoerbaar is. Aanvullende soektog-optimeringstegnieke word geïmplementeer om die berekeningskomplek-siteit van die algoritme te verlaag. Soos die aantal monsters toeneem, streef die algoritme na 'n optimale oplossing; sodoende herlei dit na 'n veilige, byna-optimale roete wat die voorspelde konflikgebied vermy. Die ontwikkeling en regverdiging van die verskillende metodes wat gebruik is om die basiese algoritme aan te pas vir die toepassing daarvan as 'n konflik-vermyding-stelsels word in diepte beskryf. Die finale stelsel word gesimuleer deur 'n vereenvoudigde vliegtuigmodel te gebruik. Die simulasie resultate dui daarop dat die voorgestelde algoritme verskeie konflikscenario's suksesvol kan oplos, insluitend die generiese tweevlieg-tuigscenario, die slegs-terreinscenario, die tweevliegtuig-met-terreinscenario en die veelvuldige vliegtuig-en-terreinscenario. Die ontwikkelde algoritme is in 'n beisge (*cluttered*), dinamiese omgewing getoets om te verseker dat dit 'n besige lughawescenario kan hanteer. 'n Statistiese ontleding van die simulasie resultate bewys dat die algoritme vinnig en betroubaar 'n aanvanklike oplossingspad kan vind, addisioneel word die oorblywende berekeningstyd ook gebruik om na 'n byna optimaleoplossing te streef.

Acknowledgments

I would like to thank Stellenbosch University and the Electronic Systems Laboratory the use of their facilities, during the commencement of this project.

A special thanks goes to my two supervisors Mr JAA. Engelbrecht and Dr CE. van Daalen for their help during my Masters. Their enthusiasm and continuous aid greatly assisted and motivated me. I am sure that without it I would not have completed the project.

Finally I would like to thank my father; Dr AM. van den Aardweg and Miss J. Roodt for proof reading the document.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations and Acronyms

2D	Two Dimensional
ACAS	Airborne Collision Avoidance System
ATC	Air Traffic Control
CAA	Civil Aviation Authority
CD&R	Conflict Detection and Resolution
CDF	Cumulative Density Function
EGPWS	Enhanced Ground Proximity Warning System
FAA	Federal Aviation Administration
FL	Flight Level
GPS	Global Positioning System
GPWS	Ground Proximity Warning System
ICAO	International Civil Aviation Organisation
IFR	Instrumental Flight Rules
LPF	Low Pass Filter
LPM	Local Planning Method
MSL	Mean Sea Level
PC	Probability of Conflict
PDF	Probability Density Function
PRM	Probabilistic Roadmap
RA	Resolution Advisory

RFCF	Runway Field Clearance Floor
RRT	Rapidly Exploring Random Tree
RRT*	RRT-star
TA	Traffic Alert
TAWS	Terrain Avoidance Warning System
TCAS	Traffic Collision Avoidance System
TCF	Terrain Clearance Floor
VFR	Visual Flight Rules

Symbol Conventions

x	Scalar
$\mathbf{x}$	Vector
X	Set
$\mathbf{X}$	Matrix
$x(\cdot)$	Function
$\dot{x}$	Derivative of x
$x^{+}(\cdot)$	Positive Function
$\hat{x}$	Estimated Value
x^{*}	Optimal Value

Greek Symbols

α	Arbitrary Angle
η	Load Factor
ω	Rate of Turn
Ω_c	Conflict Region
ϕ	Roll Angle
ψ	Heading Angle
τ	A path, representing all the states depicted by an edge
Ψ	Rotation Matrix
θ	Flight Path Angle

Lowercase Letters

$\dot{h}$	Climb rate
$\mathbf{a}, \mathbf{b}$	Arbitrary Vector
a	Semi-major axis of an Ellipse
b	Semi-minor axis of an Ellipse
d	Distance
$d_{\perp}$	Perpendicular Distance
e	Edge, where $e \in E$
f	Arbitrary function
g	Gravitional Acceleration
k_n	A single nearest neighbour to n, where $k_n \in K_n$
n	A newly sampled state in the C-space
n_S	Desired maximum number of samples
r	Radius
s	Position States
t	Time
u	Control Input
v	Vertex, where $v \in V$
xyz	Coordinates within the XYZ-axis system
k	Number of nearest neighbours determined

Subscripts

$goal$	Goal
I	Inertial Axis
$init$	Initial

Uppercase Letters

$\bar{V}$	Magnitude of the velocity
$\mathcal{C}$	Configuration space (C-space)
$\mathcal{H}$	Aribitrary Host aircraft
$\mathcal{M}$	Manoeuvre Space
$\mathcal{O}$	Aribitrary Obstacle aircraft

$\mathbf{T}_d$	Deformation Matrix
A_θ	Altitude shift manoeuvre
$C(\cdot)$	Accumulative positive Cost function / Cost to come function
$C(\cdot)$	Accumulative postive Cost function / Cost to come function
E	Set of all edges in a graph
F	Focal distance of an Ellipse
F	Forces
G	Graph structure, where G = (V,E)
K_n	A set of the k-nearest neghbours to n
L	Left turn manoeuvre
P	Successful path from the intial to the goal state
R	Right turn manoeuvre
T	Tree structure, where T = (V,E)
V	Set of all vertices in a graph
XYZ	Three dimensional orthogonal axis system
D	Drag
E	Energy
L	Lift
O	Center of a Circle
P	Power
S	Straight and Level manoeuvre
T	Thrust
W	Weight

Chapter 1

Introduction

Commercial air travel is a defining feature of the modern era and can be considered one of the major technological accomplishments of our time. Major advances in the technological capabilities of aircraft have resulted in fast, comfortable travel to nearly all corners of the globe. Current air traffic statistics stated in a release by the International Civil Aviation Organization (ICAO) that approximately 2.9 billion people made use of air transport in 2012 [15]. This current demand for air travel is substantial. Furthermore the growth in world population is expected to lead to an even higher demand for commercial air travel. This potential increase in the number of flights could lead to even more congested airspaces, especially those pertaining to airport environments. A need therefore arises for the development of more reliable systems that can ensure safety during flight.

The prediction and avoidance of potential conflicts with both terrain and other aircraft is very important to ensure safe airspaces. A conflict is defined as a breach of the protected airspace zone[1] surrounding each aircraft. The prediction of potential conflicts is managed by a conflict detection system, once a conflict is detected a conflict resolution system determines the actions required to avoid the predicted conflict. The collaboration of these systems is of paramount importance to ensure safe airspaces. The existing conflict avoidance systems, namely Air Traffic Control (ATC), the Traffic Collision Avoidance System (TCAS) and the Enhanced Ground Proximity Warning System (EGPWS), are very efficient and effective, but these systems do have some limitations as well as the potential for improvement. ATC is managed by a human controller and therefore subject to human error. TCAS is a rule-based system that is effective and robust in scenarios for which it was designed, but scenarios for which it was not designed could pose problems. Other limiting factors include the possible miscommunication between ATC and TCAS, due to a lack of integration between the systems as well as TCAS's lack of horizontal resolution.

The need for a more reliable conflict avoidance system instigated the establishment of the RoCRAE project by Airbus. This project involves research into alternative conflict detection and resolution methods. The project aims to contribute to a safer airspace by developing a complete conflict avoidance package that is capable of detecting and resolving conflict within airport environments. The research for this avoidance system was divided into two Masters projects: one involving the detection of conflict and the other the resolution of

[1] A description of the protected airspace zone is shown in Section 2.1.1

conflict. This thesis describes the development of a conflict resolution system to be implemented with an external detection module that would ultimately function as a complete avoidance system for an aircraft. The method proposed in this thesis presents a robust, sampling-based path-planning algorithm for implementation on commercial airliners. The system makes use of numerous external functions including a rejection sampler, a dynamic sampling space, intricate local planning methods as well as various optimisation techniques to ensure that the paths generated are flyable and converge towards an optimal solution. The proposed system aims to perform complex simultaneous aircraft conflict and terrain avoidance within cluttered dynamic environments.

1.1 Problem Statement

The goal of this research is to develop a conflict resolution module for commercial aircraft that is able to simultaneously resolve predicted conflicts with both intruder aircraft and terrain. The role of the conflict resolution system is to perform path planning and to generate a safe, flyable path for the host[2] aircraft that avoids the conflict region while adhering to the dynamic constraints of the aircraft. The path planning algorithm must be able to perform in real-time, so that the resolution manoeuvres can be executed early enough to avoid the conflict region. The primary goal of the system is to determine a safe flyable path that successfully resolves for the predicted conflict. Once a safe path has been determined, the system must utilise the remaining computation time to optimise the resolution path based on predefined metrics. The proposed resolution system should adhere to the requirements listed below to ensure that these goals are achieved.

1. Given an initial and goal state the algorithm will determine a path that connects these states[3].
2. The system must ensure that this path is free of conflict, through the use of an external conflict detection module.
3. The path determined must also be flyable. This implies that the path must abide by the dynamic constraints[4] of the aircraft.
4. Once a path is determined the system should attempt to optimise the path based on predefined metrics.
5. The system should ultimately be applicable to uncertain, dynamic and cluttered environments.
6. The application of the system to real world problems requires real or near real-time computation.
7. Finally the algorithm must ensure that if a path exists it will be determined.

It is assumed that the conflict resolution system is used in conjunction with an external conflict resolution module that predicts the probability of a conflict along the proposed path. The development of this conflict detection system is outside the scope of this project, and is the subject of another master's degree project by Pienaar [16]. The operation and performance of the conflict resolution module must be verified in test scenarios that involve a cluttered and dynamic environment, involving multiple intruder aircraft and terrain, such as in the airspace around an airport.

[2]The resolution system is functional on the host aircraft, while all other aircraft are assumed to continue on their current trajectories and apply no resolution actions.

[3]A state is defined in this thesis as the physical states of the aircraft. See Section 3.1.2 for the states composition.

[4]Dynamic or differential constraints refer to the physical capabilities of the vehicle. See Section 3.1.4 for details.

1.2 Overview of Proposed Solution

The sampling-based path planning algorithm selected as the conflict resolution system was derived from the existing research in the field of aircraft conflict resolution and path planning. This sampling-based system is capable of determining paths in dynamic and static environments, thereby combining the functionality of TCAS and the EGPWS, additionally the proposed system is capable of generating both vertical and horizontal paths.

The functioning of the proposed resolution system can be seen in Figure 1.1. The figure highlights the interaction between the resolution and detection modules.The developed algorithm, located within the conflict

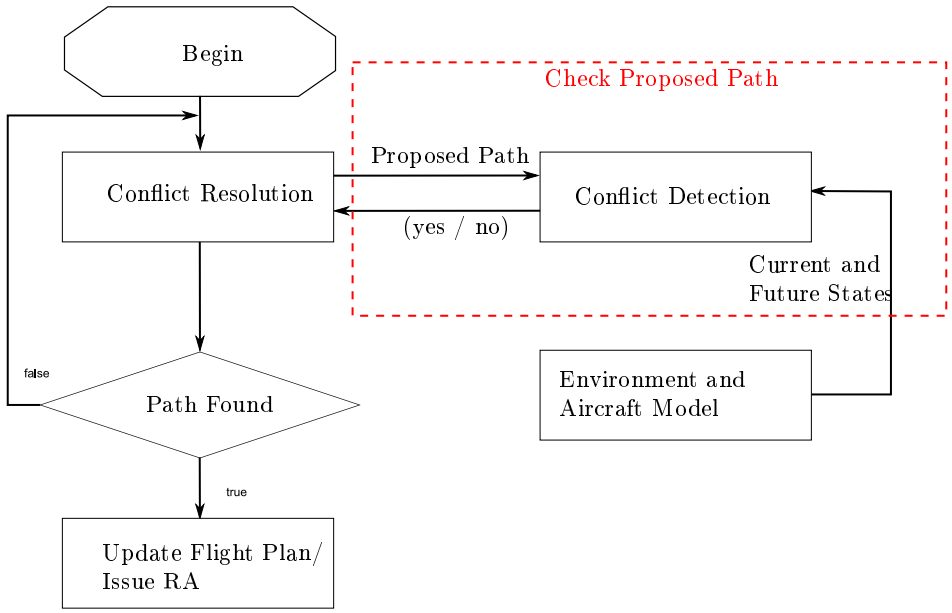


Figure 1.1: A flow diagram depicting the interaction between the conflict resolution and detection modules

resolution block in Figure 1.1, randomly samples the search space, these samples are connected using a local planning method to ensure that the connecting paths are indeed flyable. Each flyable path is tested using an external conflict detector, as shown in Figure 1.1, to ensure that the proposed paths are conflict-free. The external conflict detection module has access to a model of both the environment and the aircraft. Once a flyable, conflict-free path to the goal has been determined the algorithm applies various search optimisation techniques to reduce the computation time and promote the convergence towards an optimal solution. The final resolution path is conveyed to either the human pilot or the autopilot system as a set of reference commands required to follow the proposed path. The resultant system adheres to the requirements stipulated in the section above. The selected algorithm as well as the additional methods implemented ensures that the system determines a flyable conflict-free path fast and effectively. The development of this robust resolution system is described in this document as well as the methods required to cater for uncertain, cluttered and dynamic environments. Additionally the management and integration of the external conflict detection system is explained. Finally the system is applied to four challenging conflict scenarios, including: the generic two aircraft scenario, a terrain only scenario, a two aircraft with terrain scenario and a multiple aircraft with terrain scenario. An analysis of the results gleaned from the different tests scenarios is conducted to determine the applicability of the proposed system as an aircraft conflict resolution system.

1.3 Structure of Document

This thesis aims to explain the development of the above-mentioned conflict resolution system as well as all relevant background information required to justify the development of the system. Finally the thesis verifies the developed conflict resolution module through the simulation of multiple scenarios, with varying complexity, to ensure that the proposed system is applicable as a potential conflict resolution system. This document is structured so that initially a broad overview and background of the problem is described. Once the general methodologies and terminologies have been established the document focuses on the selection and implementation of the proposed conflict resolution algorithm.

Chapter 2: Literature Review

This chapter describes the constraints placed upon the required resolution system. Additionally a broad overview of the existing resolution systems and research is given. From this the most promising family of path planing algorithms are selected for further research. A description of what a conflict detection and resolution system entails is depicted in Section 2.1. Background descriptions of the existing resolution system used can be found in Section 2.2, while an analysis of the currently research resolution techniques, that have not been implemented, is discussed in Section 2.3.

Chapter 3: Modeling of System and Environment

The modelling of both the terrain and aircraft are described, these models ensure that the applied system can function fast and effectively. The simplified models of the aircraft and environment used by the system are described in Sections 3.1 and 3.2. Additionally Section 3.1 helps describe the potential limitations placed on the aircraft that have to be adhered to, to ensure flyable paths. The methods used to manage uncertainty are found in Section 3.2.3.

Chapter 4: Sampling-Based Algorithms

The selected family of path planing algorithms are discussed in depth and the most promising methods are highlighted. This research is used to select a sampling-based algorithm as a basis for the implemented resolution system. This chapter discusses the validity of sampling-based algorithms as a potential conflict resolution system. The concepts and functionality of general sampling-based algorithms are found in Sections 4.1 and 4.2, while an overview of the different search algorithms is given in Section A. The three prominent sampling-based techniques applicable to aircraft scenarios are explained in Sections 4.5, 4.6 and 4.7.

Chapter 5: System Implementation

This chapter discusses the adaptations made to the algorithm highlighted in Chapter 4 to ensure its effective implementation as an aircraft resolution system. The practical complexities, design choices and assumptions are highlighted in Section 5.2. Section 5.4 depicts the interaction between the conflict resolution module and the external detection module. The functionality of the resolution algorithm selected is described in depth in Section 5.3. The development and selection of the sampling strategies are described in Section 5.5, while Section 5.6 discusses the methods used to ensure that all paths are flyable. In Section 5.7 we focus on

the optimisation of the path based on different metrics. The chapter is concluded with Section 5.8, which discusses techniques that have been implemented to optimise the computation of the paths.

Chapter 6: Simulation and Results

The implementation of numerous test scenarios are described and analysed in this chapter. The tests aim to determine the viability of the proposed system as a conflict resolution module. The chapter describes the different conflict scenarios tested as well as the effects of the different cost functions. An analysis of the algorithm's ability to derive an initial path can be seen in Section 6.2.2, while the convergence rate of the algorithm to an optimal solution is discussed in Section 6.2.3. Additional tests are executed to determine the effects of the different techniques and adaptations applied to the system.

Chapter 2

Literature Review

This chapter focuses on the relevant literature required to understand Conflict Detection and Resolution (CD&R) systems. We give an overview of the general CD&R system as well as a description of how the environment affects the system. In order to understand how a CD&R system works we first look at the existing systems that are used on commercial aircraft as well as the research being done on new alternative resolution methods. Finally, a description of the environment and the requirements of the system to cater for this environment are discussed.

We start by defining what exactly is implied by the term "conflict" in the context of an aerospace application in Section 2.1.1. Thereafter we give an overview of what a conflict detector (Section 2.1.2) and a conflict resolver (Section 2.1.3) are and what is expected of these systems. In Section 2.2 we move on to the CD&R systems currently implemented on commercial airliners, discussing their functionality, advantages and limitations. This is followed by a description of the research being done on alternative resolution methods in Section 2.3. Here we discuss the different approaches, highlighting the advantages and possible shortcomings of the different systems. Finally a description of the airport environment, its restrictions, regulations and possible challenges is supplied in Section 2.4.

2.1 Conflict Detection and Resolution System

This section explains the basic concepts that are required to understand what a CD&R system entails. The system is made up of two distinct, separate modules, namely: the Conflict Detector and the Conflict Resolver. These two modules function independently, but are coupled. The effectiveness and efficiency of the system are dependent not only on the individual modules but on the interaction between these modules. The typical functioning of a CD&R system is that when a conflict is detected, the resolution module is activated and determines a conflict free path. Sections 3.2.3 and 5.4 describe this interaction in depth.

In order to understand how a CD&R system functions, we have to clarify certain terms. The sections that follow define what we mean by the term "conflict" and then describes the basic functionality and different aspects of general conflict detection and resolution modules.

2.1.1 Definition of Conflict

In order to understand the concept of an aircraft conflict resolution system, we first have to understand what a conflict is. A conflict can be described as a breach of the protected airspace zone that surrounds each aircraft. When another aircraft enters the host aircraft's protected airspace zone, then we say a conflict has occurred.

The protected airspace zone (protected zone) is defined by the minimum separation standards based on the aircraft's current phase of flight. The En Route, free flight protected zone is defined as a cylindrical space surrounding the aircraft that has a radius of 5 nm (nautical miles) $\approx$ 10 km and a height of 2000 ft (foot) $\approx$ 600 m, as shown in Figure 2.1 [17, 18, 1]. Here free flight implies that an aircraft can fly its own preferred trajectory. When an aircraft approaches an airport the shape of the protected airspace zone changes.

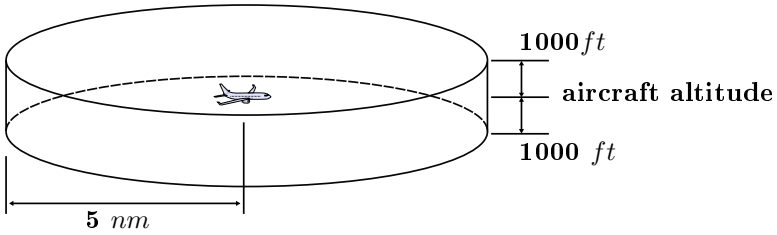


Figure 2.1: En-route protected airspace zone, adapted from Hoekstra et al. [1] (Vertical scale exaggerated)

The protected zone defined near or within an aerodrome (airport or airfield) according to the ICAO is not a set shape or distance, but is defined with respect to the current state or configuration of the aircraft [19]. Note that the state of the aircraft refers to the physical aircraft states (for details refer to Section 3.1). The vertical separation is defined as 2000 ft above FL 280[1] ($\approx$ 28000 ft) and 1000 ft below it. The horizontal separation is grouped into three categories: 5 , 10 and 15 minutes respectively. Different situations require different separation classifications. These situations are described in depth in Chapter 5.4 of the Procedures for Air Navigation Services: Air Traffic Management by the ICAO [19].

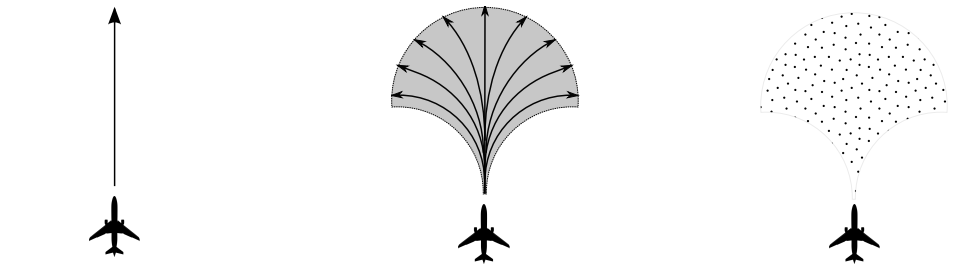
The conflict region can therefore be breached while the aircraft remain relatively far apart. This means that even though a conflict has occurred, a collision can still be avoided. Thus, if all conflicts are avoided, any possible collisions will be prevented.

2.1.2 Conflict Detection

Implementation of an effective conflict resolution system requires a reliable detection module to ensure that the desired paths are indeed conflict free. A conflict detection module first estimates the future states of all aircraft (host and obstacle/s) in a set region, then through the application of predefined metrics determines whether a conflict will occur. These metrics can vary in complexity ranging from a simple single parameter metric (distance) to complex multiple parameter metrics (distance, time, velocity, manoeuvrings) [17].

[1] Flight Level (FL) is the barometric pressure altitude of the aircraft, expressed in hundreds of feet. It is determined by comparing the local pressure to the standardised sea-level pressure of 1013.25 hPa (29.92 Hg). It is not necessarily the same as the aircraft physical altitude [19].

A conflict detector is only as accurate as its ability to model the predicted or estimated future states of the system, thus state propagation of the elements in the system dictates the validity of the conflict detector. There are three main propagation techniques, highlighted by Kuchar and Yang, that are commonly used to model the future trajectories of the aircraft [18]. These are the nominal trajectory method, worst case approach, and probabilistic analysis of states as seen in Figure 2.2.



(a) Nominal Approach (b) Worst Case Approach (c) Probabilistic Approach

Figure 2.2: State propagation methods, adapted from Kuchar and Yang [2]

The nominal trajectory method, Figure 2.2a, models all future aircraft states as a single path, based on the aircraft's current velocity vector. This is the most simplistic method and requires the least computation time, but does not cater for uncertainty and will result in problems if any sensor errors occur or the aircraft deviate from their predicted path.

The worst case approach, in Figure 2.2b, is the most computationally expensive method. It considers every possible trajectory that the aircraft can fly, bounded only by the aircraft's dynamic capabilities. Additionally the worst case approach is over conservative resulting in many false alarms. This diminishes the integrity of the system and can cause a pilot to ignore predicted conflicts. Both the above-mentioned conflict detectors have one characteristic in common: each outputs a binary result [1 or 0] (hit or miss) for the detection of conflict.

The probabilistic approach, seen in Figure 2.2c, is essentially the middle ground between the previous two methods, weighing the probability of conflict between 0 and 1 within the dynamically flyable region. The previous approaches can be considered special cases or subsets of the probabilistic method [20, 18, 2]. The probabilistic approach is a balance between the two above mentioned methods. It is more conservative than the nominal trajectory method ensuring that uncertainty in the environment is catered for. While remaining less conservative than the worst case approach thereby reducing the chance of false alarms. The main advantage of the probabilistic method is that it is computationally viable and caters for uncertainty along the nominal paths.

2.1.3 Conflict Resolution

As discussed in Section 2.1.2 conflict detection is the process of deciding if certain actions should be taken (i.e. when a conflict is predicted), while conflict resolution involves determining how or what action can be taken to avoid the conflict, before it will occur [18].

2.1.3.1 Resolution Systems

The efficiency of the conflict resolution module is directly linked to the conflict detector's ability to convey reliable and accurate predictions. This results in different systems of conflict resolution based on the type of detection information available. One such distinction is the look ahead time, because this directly affects the reliability of the conflict detector's predictions. We cater for it by making use of different conflict resolution systems for different look ahead times.

1. **Long term**

   Long term conflict resolution, also known as Flow or Trajectory Management can be considered a form of Flight Planning. This is because the conflict horizon (time to conflict) is a matter of hours or days, therefore the level of uncertainty is so high that it is not really possible to perform any acts of resolution. This form of conflict resolution can be considered more of a preventative measure to ensure minimal potential conflict scenarios [12].

2. **Mid term**

   Mid term conflict resolution attempts to prevent any infringement of the aircraft's protected airspace zone. This method has a conflict horizon in tens of minutes, giving the resolution module time to resolve any potential conflicts [12].

3. **Short term**

   Short term conflict resolution is the last preventative measure and is only implemented if a conflict is pending despite all other detection and resolution techniques' attempts to resolve it. Here the conflict horizon is a matter of minutes. Currently Air Traffic Control (ATC) is responsible for detecting and resolving these forms of conflict, while on board systems and crew with the aid of the Traffic Collision Avoidance System (TCAS) handle collisions that are a minute or less away [12]. Here the only criterion is a fast successful response.

2.1.3.2 Manoeuvres

When a conflict has been detected the resolution module determines a safe path that will avoid the predicted conflict. This path determined is described by a set of manoeuvres, where a manoeuvre refers to any dynamically realisable action applied over a period of time (see Section 5.6.1 and 5.6 for details). By describing a path as a sequence of manoeuvres it makes the development of the resolution paths tractable, since there exist fewer choices to be made, when compared to describing a complex trajectory as a set of points. The only constraint placed upon any manoeuvre or manoeuvre sequence is that they adhere to the differential constraints of the aircraft, described in Section 3.1.4 [21]. This means that no manoeuvre may be considered if it violates the aerodynamic capabilities of the aircraft, or exceeds the maximum loads that can be endured by the aircraft structure or passengers. The manoeuvres that are generally used for conflict resolution can be grouped into three main categories:

1. Horizontal,

2. Vertical, and

3. Velocity manoeuvres.

Horizontal manoeuvres are defined as a combination of turns, where each turn can be described using either bank angles or angular rates. Vertical manoeuvres result in changes in altitude by applying either ascend or descend commands. Velocity manoeuvres are changes in the aircraft speed. These three different manoeuvre types can be used individually, consecutively or in combination to avoid potential conflicts.

Most conflict scenarios will only require a single manoeuvre type, while more complicated scenarios may require a combination. Of these manoeuvres, vertical or altitude changes are considered to be the most economical (based on fuel consumption and time), followed by heading changes (horizontal) and finally speed changes [22]. We can possibly attribute this ranking to the shape of the protected airspace zone, since the vertical separation standards are less stringent than the horizontal requirements. This becomes clear when we consider that the clearance required by a vertical manoeuvre is 30 times less demanding (1000 ft versus 5 nm) [22].

2.1.3.3 Centralised and Decentralised resolution

Conflict resolution can be managed by two distinct systems: either by the ground station (centralised) or by each individual aircraft (decentralised). The centralised approach has one master controller that has complete knowledge of the states of all the aircraft in the system, which is generally based at a ground station (ATC). With this information it attempts to achieve a global objective. On the other hand the decentralised approach is where the system is active on each individual aircraft (TCAS) and therefore has limited knowledge of the other aircraft. This method attempts to achieve a unique objective based on the requirements of the host aircraft (aircraft on which the system is active). What this means is that when a conflict is detected the centralised approach will find resolution paths that yield an effective global solution, while the decentralised approach will find the most effective local solution [12]. The main differences between the two approaches are highlighted in Table 2.1.

Table 2.1: Comparison between Centralised and Decentralised Resolution Management [12]

Decentralised Management	Centralised Management
Independently manoeuvres its own host aircraft	Has authority to dictate the manoeuvring of each aircraft in the system
Determines an optimal path (locally) that has been defined by its own stakeholders	Optimises the global situation (for all aircraft)
Receives most up-to-date information regarding its own future states	Receives global information regarding aircraft state and planned flight path.
Has to solve a small subset of the global traffic situation	Handles a large number of aircraft
Limited computational power	

2.2 Existing Resolution Methods

This section describes the resolution systems that are currently being implemented on commercial airliners, collectively know as the Airborne Collision Avoidance System (ACAS). Each system described below manages the conflict resolution for different phases and aspects of flight: ATC handles CD&R and traffic management near or in airport environments, TCAS resolves dynamic airborne (aircraft to aircraft) collisions and the Enhanced Ground Proximity Warning System (EGPWS) caters for potential terrain collisions. The methods are all independent of each other and have separate detection and resolution systems. The interaction of these methods is defined by specific regulations in order to prevent possible miscommunication.

The automated resolution systems TCAS and EGPWS currently act as advisories and warnings to the pilot, therefore no resolution control is executed by the systems [23, 24]. The systems cater for near or short term collision avoidance, providing alerts 15 to 55 seconds in advance [25]. Long and mid term collision avoidance is generally managed by the on board crew in collaboration with the ATC [26]. These methods have been tried and tested and adhere to the rigorous safety standards set forth by the Federal Aviation Administration (FAA) and ICAO. This being said, there are notable instances where miscommunication between the systems has resulted in near air conflicts and even collisions. Examples of these are: Yaizu (Japan) mid-air conflict January 2011 and the Überlingen (Germany) mid-air collision in July 2002. In both cases one aircraft followed the TCAS advisories while the other followed the ATC instruction [27].

2.2.1 Air Traffic Control

The ATC is a centralised, human based control unit, located within airports. There are numerous ATC stations and the task of managing specific regions of the aerodrome is assigned to each station. The roles of the ATC service are to prevent potential aircraft collisions and to maintain an orderly flow of traffic [28]. In order to achieve these goals the ATC requires all relevant aircraft data: current airspeed, estimated arrival times, flight plan and all deviations from the planned track [28]. Furthermore all information pertaining to the intended manoeuvres of the aircraft as well as variations therein must be provided [19]. Using this the ATC can manage its required airspace and facilitate co-ordination between the aircraft. ATC commands have to be adhered to when any aircraft enters airspace classes A to E surrounding the aerodrome (see Section 2.4.2 for details). All additional regulations pertaining to the control limits of the ATC and when this control takes effect can be noted in the ICAO Procedures for Air Navigation [19] and Annex 2 of the International Standards [28].

The ATC requires all relevant data pertaining to the aircraft within the aerodrome. If a potential collision is detected the ATC attempts to resolve it by determining a globally optimal solution that resolves the collision. The resultant resolution manoeuvres are determined by the human controllers through the aid of computers, therefore the ATC system can be subject to human error.

2.2.2 Traffic Collision Avoidance System

TCAS is an optimal, rule based system that handles dynamic, aircraft to aircraft CD&R [18]. TCAS is a decentralised CD&R system located on all commercial aircraft. The system functions independently from

the ATC. When a TCAS advisory is issued the pilot is required to act on that advisory. If a contradictory advisory is issued by the ATC, the TCAS advisory should take precedence and the pilot should notify the ATC. The potential for miscommunication between TCAS and ATC has posed problems in the past.

2.2.2.1 Development

TCAS I started as a traffic alert (TA) system only, which detected whether traffic is present in the vicinity of the aircraft and issued an alert if possible collisions could occur. No resolution advisories were determined. Advances made to TCAS II & III incorporated resolution advisories (RA). This occurs when an obstacle aircraft enters a predefined protected region around the host aircraft. RA issue a command that will resolve the potential conflict. The commands issued are expressed as vertical manoeuvres. The manoeuvre set is made up of up and down (climb or descend) commands with varying strengths. Additionally TCAS II implemented collaborative resolution between systems, implying that all aircraft equipped with TCAS can communicate and resolve the predicted conflict together.

The implementation of TCAS was accomplished due to advances in transponder technology and requires Mode A/C or Mode S transponders to function. Using these, the aircraft is able to request the altitude data from all the obstacle aircraft in range and determine its estimated distance from the obstacles using the round time of the transmission [25, 27]. TCAS requires this data every second and from it can calculate the estimated range, range rate, altitude, altitude rate and bearing of all obstacle aircraft. Using this, it is possible to predict the path of all aircraft and determine if a collision will occur. Improvements made in TCAS III added horizontal resolution, but due to large uncertainties in the miss distance estimates, based on bad bearing estimation, it was deemed too unreliable for commercial use [29, 30].

2.2.2.2 Functionality

This section describes the basic functionality of the TCAS system. TCAS makes use of a predetermined set of rules as well as coordination between all aircraft in the system to determine a set of globally optimal manoeuvres for each affected aircraft. These manoeuvres are limited to climb and descend commands with certain strengths or magnitudes. The manoeuvre strengths vary depending on what is required to resolve the conflict. The FAA state that a typical example of a climb or descend command is approximately 1500 ft/min, while an increase climb or descend command typically requires 2500 ft/min [3].

Figure 2.3 is a graphical representation of the RA and TA alert regions and their typical look ahead times. Note that the regions' times vary depending on the aircraft altitude. When an obstacle aircraft enters the TA or RA region, TCAS will issue a traffic or resolution alert to the pilot. The TA and RA commands are determined based on the track of the host and obstacle aircraft. A typical example is that the aircraft at a lower altitude will be requested to descend while the other aircraft will be required to ascend. This ensures that the safe separation zone is not breached. Figure 2.4 depicts the expected pilot response to an oncoming obstacle aircraft at a lower altitude. During the TA phase no deviation is expected; this alert is only to raise awareness of the possible conflict. When the RA is issued the pilot has to respond within 5 seconds of the command, following the response accurately. When the threat has been averted the pilot returns the aircraft to its original flight path.

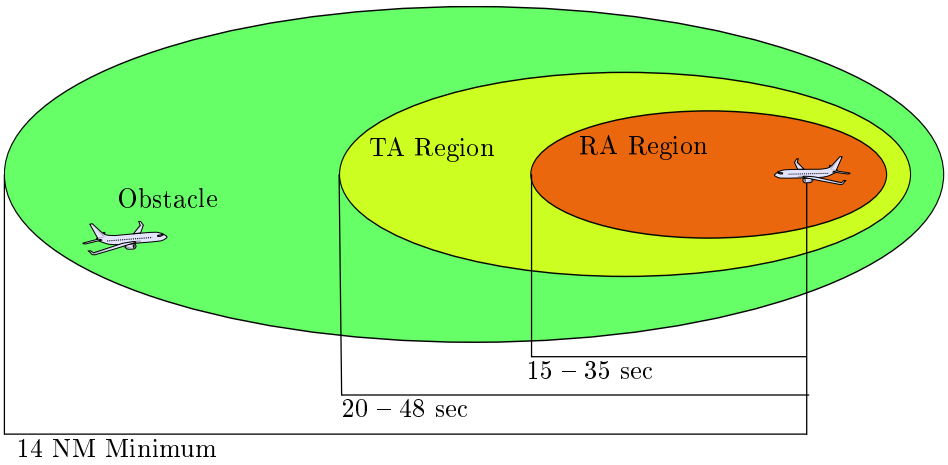


Figure 2.3: TCAS, RA and TA alert times, adapted from the FAA presentation [3]

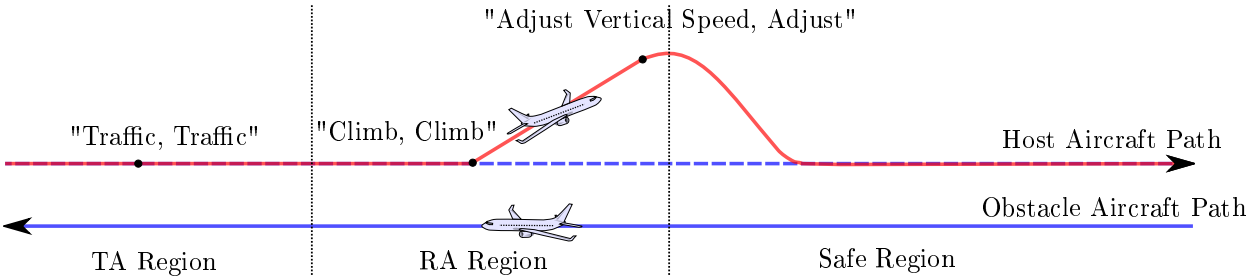


Figure 2.4: Expected TCAS pilot response, adapted from the FAA presentation [3]

2.2.2.3 Limitations

The current implementations of TCAS are subject to certain limitations. The limitations include those placed upon the system to ensure safe manoeuvres and to prevent potential conflicts. The limitations of the current system are listed below [25, 27].

1. TCAS only provides RA and TA with regard to obstacle aircraft equipped with a working TCAS.
2. The system will fail if communication with the barometric altimeter, radio altimeter or transponder (Mode S or A/C) is lost.
3. No RA is given to an aircraft with a vertical rate larger than 10 000 ft/min (50 m/s) or with closure rates of 1200 knots (620 m/s)
4. Descent and increase descent rate RA's will be limited if the aircraft altitude is below 1100 ft (335 m) and 1550 ft (475 m), respectively. Below 1000 ft (300 m) all resolution advisories are inhibited.
5. Due to inadequate bearing information current implementations of TCAS do not support horizontal resolution [25]. Burgess stated that the lack of horizontal separation is a hindrance to the current system [29].
6. Due to the implementation of different CD&R systems the EGPWS, Terrain Avoidance Warning System (TWAS) and wind shear warnings take precedence over TCAS advisories [25, 27].

7. The success of TCAS resolution relies heavily on the response of the pilot in executing the required manoeuvres correctly and promptly.

Some of the limitations listed above are unavoidable physical requirements (1, 2), while others are safety limitations (3, 4). Finally there are some limitations that could possibly be reduced (5, 6) with advances in technology. It must be noted that the current implementation of TCAS has been tested on dense airspaces at high speeds and have recorded positive results. Tests have concluded that TCAS is effective with up to 30 aircraft at speeds of 260 m/s. This makes it an effective and viable system, although there is room for improvement with regard to horizontal resolution and integration between multiple systems.

2.2.3 Ground Proximity Warning System

The Ground Proximity Warning System (GPWS) is a terrain avoidance system that warns the pilot when a terrain collision is imminent. The GPWS functions similarly to TCAS in the sense that it is a decentralised system located on the host aircraft. The GPWS differs from TCAS with regard to conflict detection; it only caters for static obstacles (terrain) and it functions only as a warning system. Therefore no resolution manoeuvres are determined. The pilot is only warned that a terrain collision has been predicted based on the current trajectory and altitude. The original system made use of radar to determine the current altitude and warn the pilot if the aircraft is flying too low. This system was subject to a major flaw: only the current altitude was used, therefore steep increases in terrain could not be predicted, resulting in possible conflicts. Advances in the GPWS yielded the EGPWS. This updated system makes use of radar, the Global Positioning System (GPS) and an obstacle, terrain and runway database to ensure improved safety [23, 31].

2.2.3.1 Enhanced Ground Proximity Warning System

The EGPWS makes use of multiple modes of warning that cater for different scenarios. These modes range from 1 to 7, some notable alerts include rising terrain and insufficient terrain clearance. The rising terrain alert is based on the current radio altimeter readings and is issued when a steady decrease in altitude is detected. This alert's reliability is improved due to the use of the terrain look ahead function that uses the aircraft position, flight path angle, track, speed and a terrain database. Using this, the EGPWS predicts whether a potential collision will occur and issues the required alerts. The insufficient terrain clearance is essentially the functionality of original GPWS, and issues an alert if the aircraft drops below an altitude of 1000 ft [31]. Notable advances in aerodrome functionality include the Terrain Clearance Floor (TCF) and Runway Field Clearance Floor (RFCF) functions. The TCF function alerts the pilot of a possible premature landing or descent. This is determined by using the aircraft altitude and the distance to the centre of the runway to determine a safety TCF altitude around the runway. The RFCF is used when the approach runway is higher than the surrounding terrain. It determines a minimum approach profile and alerts the pilot if this profile is breached [31]. Current implementations of the EGPWS only provide warnings to the pilot, requiring the pilot to manually resolve the conflict.

2.3 Existing Conflict Resolution Research

This section gives an overview of existing research on conflict resolution and path planning methods that have not been implemented in commercial flight. Over time numerous different conflict resolution or motion planning approaches have been researched. Kuchar and Yang compiled two studies on the different resolution methods and recorded over 60 alternative methods of conflict resolution [2, 18]. They categorised each of the resolution methods as follows: Force Field, Optimised, Prescribed or Rule Based and Manual.

The aim of a resolution algorithm is to determine a resolution path to a known goal position that will avoid the conflict regions. Numerous techniques have been developed that are not applicable to an aircraft scenario. We have selected a few promising applicable techniques and will discuss these methods in Sections 2.3.1, 2.3.2, 2.3.3 and 2.3.4 below.

2.3.1 Force Field Methods

The force field method is the application of virtual electrostatic forces on conflict resolution scenarios. This method attempts to apply general electrostatic equations to a conflict scenario to determine a globally optimal solution to the problem. These methods have been widely used for motion control of mobile robots, although they are not very popular as a CD&R system for an aircraft. This is because of the local minima problem that can theoretically arise in any gradient descent algorithm, unfortunately force field methods are part of this class of algorithms [9]. This means that force field methods cannot guarantee that a safe resolution path will be found, posing a problem with regard to practical applications. Additionally, when attempting to determine a feasible solution with bounded inputs it is possible that the force field methods can not find a path that abides by the dynamic constraints of the vehicle (see Section 3.1.4 for detail on dynamic constraints) [12]. This section discusses three different applications of the force field method that attempt to overcome these limitations.

2.3.1.1 Voltage Potential Field Method

The voltage potential field method is the original force field method applied to a practical aircraft scenario. All aircraft are modelled as charged particles and make use of adapted electrostatic equations to determine the required resolution manoeuvres [18]. Each aircraft is modelled as a positive charge while the desired goal state is modelled as a negative charge [1]. Therefore all the aircraft repel each other while the host aircraft is attracted to its goal state. These forces define the manoeuvres required to avoid the conflict. This is graphically represented in Figure 2.5. One idealistic aspect of the methods is that the system will continuously be resolving conflicts, therefore potential conflicts can be avoided well in advance using minor manoeuvres [18].

Hoesktra and Eby argue that the major disadvantage of the basic potential field method lies in its simplicity [1, 4]. Eby highlights that one drawback is that the destination (goal) will always be further away than the obstacle aircraft, resulting in the repulsive forces of the obstacle aircraft always outweighing the attractive force of the goal. This could lead to unnecessarily large deviations from the path. However, the major flaw

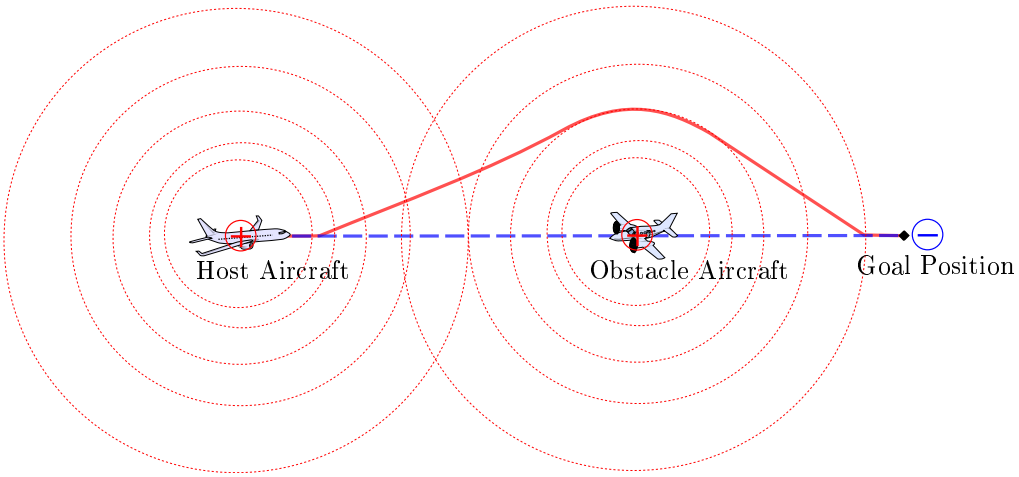


Figure 2.5: Potential Field creation and resolution manoeuvre

in the algorithm is that it can not guarantee a set safe separation distance. The separation is determined as a function of the velocities reducing as the aircraft speeds increase [1, 4]. This relationship is not viable for the implementation of a practical conflict resolution system. A more robust, reliable system is required to ensure safety.

2.3.1.2 Updated Potential Field Method by Lincoln Laboratories

Lincoln Laboratories developed an adaptation of the voltage potential field algorithm described above. This adaptation made use of the basic potential field features, but used a more pragmatic approach to solve conflicts [1]. The one major change is that the potential field-like repulsive forces are applied at the the predicted conflict's location and not around the current position of the host and obstacle aircraft. This method does not apply the physical electrostatic equations as in the potential field methods, but the equations that are applied exhibit the same characteristics. This means that the strength of the avoidance "force" is inversely proportional to the distance between the predicted obstacle and host aircraft positions.

The following is an explanation of the algorithm graphically illustrated in Figure 2.6, derived from descriptions by Hoekstra [1] and Eby [4].

1. The algorithm determines the predicted future states (positions) of both the host and obstacle aircraft (dots on the blue lines in Figure 2.6). From these future states the minimum miss distance can be determined. This is the distance at which the predicted states are closest to one another.

2. At the minimum miss distance position we determine the miss distance vector, $\mathbf{d}(t)$, which is a vector projected from the future obstacle's position towards the future host position.

3. The avoidance vector, $\mathbf{a}(t)$ is calculated from the projected host state towards the edge of the obstacle's protected zone, in the direction of $\mathbf{d}(t)$. The length of $\mathbf{a}(t)$ dictates the magnitude of the resolution manoeuvre required and therefore reflects the severity of the conflict. It is also the shortest euclidean distance out of the conflict region.

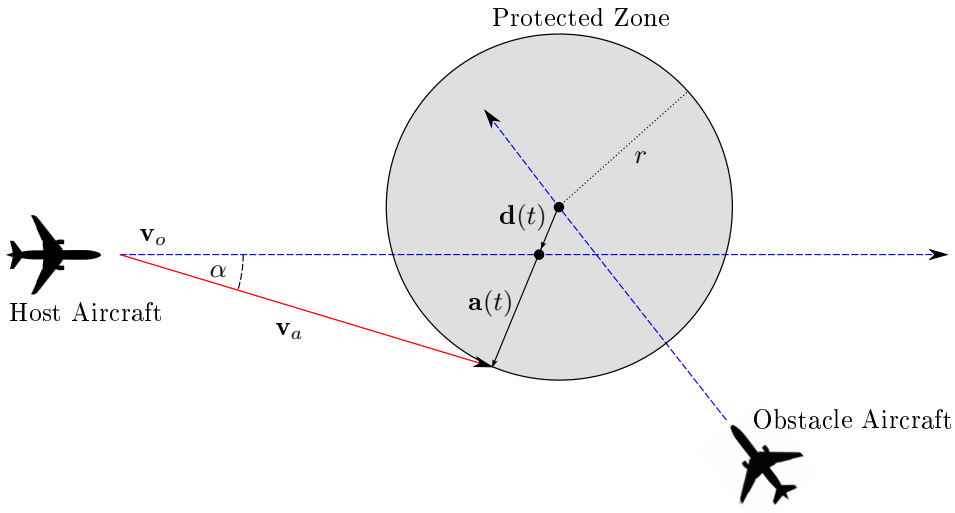


Figure 2.6: Lincoln Laboratories adaptation of the Potential Field Resolution method, adapted from Hoekstra [1] and Eby [4].

4. The deviation velocity vector is determined by dividing **a**(t) with the time to the conflict. The summation of this deviation velocity vector and the original velocity vector,$\mathbf{v}_o$, results in the advised velocity vector, $\mathbf{v}_a$, required to avoid the predicted conflict.

5. For situations with multiple aircraft, all avoidance vectors are summed and from this, the net deviation velocity is determined.

Some advantageous features of the system include the following: in the absence of conflict no change to the path occurs; conflicts far away result in little to no deviation, while conflicts close by result in large changes; and each aircraft acts as if the other will not, but if the other aircraft reacts the algorithm accounts for it. For practical implementation some additions must be added to the system. A set of rules is added to cater for possible singularities that can occur. For example if a perfect head on collision occurs then **d**(t) will be zero, causing an incorrect avoidance vector. Hoekstra et al. concluded that although this method seems promising, the Modified Voltage Potential Method described by Martin Eby [4] is a better fit for aircraft conflict resolution [1].

2.3.1.3 Modified Voltage Potential Field Method

This force field based resolution method is an adaptation of the basic Lincoln Laboratories method, by Martin Eby [4]. The basic functionality of the algorithm is the same as in the original method. Some additions have been added to the algorithm to make it applicable to a practical aircraft scenario. Eby noted that although the original algorithm worked well, some problems occurred that have to be addressed. This included: (1) in complex scenarios the paths calculated were too complicated to be carried out by a human pilot, (2) the safe separation minimum (set at 5 miles) was not strictly adhered to and (3) a few outliers had minimum separations at only 70 to 80 % of the desired separation distance.

This led Eby to apply certain solutions to prevent or prohibit these problems. In order to simplify the paths a linearisation algorithm was applied to the determined paths. This simplified the original output, which

consisted of many way-points, to a single way-point linearised path. With regard to the failures Eby noted two possible reasons that caused them. The one was a simulation error, where some aircraft were added later and in locations where resolution was near impossible, causing the infringement of the safe separation area. This can be ignored since in practice such scenarios cannot occur. The other reason is that if two obstacle aircraft approached for a head on collision, each located an equal distance from the host's projected path, then the avoidance vectors calculated by the host will cancel out, making the host aircraft fly between the two oncoming intruders.

The algorithm described was determined as a practical aid for the ATC to be used as a guide for the human controllers to determine more accurate and optimum safe paths. The system was improved and tested rigorously under sub-optimal conditions on complex scenarios, by Eby and Kelly [32]. They concluded that it represented a robust resolution system, but agreed that as the difficulty or complexity of the scenarios increased, the system exhibited a reduction of separation between aircraft. Finally they stated that in all the simulations the system successfully succeeded in resolving all conflicts. The modified voltage potential method seems to be a viable conflict resolution technique, but the theoretical limitation that a local minima problem can occur is still present. Additionally the force field method could pose computational problems as it theoretically has no bounded search region and therefore requires an unlimited look ahead time [22]. It was also concluded that this method preforms adequately, but has one drawback: causing unnecessarily aggressive manoeuvres which result in a loss of passenger comfort or could even exceed the physical limitations of the aircraft [1, 22]. Finally the algorithm's inability to place a strict bound on the conflict region is a large limiting factor, and if the possibility of a loss of separation exists it can not be applied as a commercially used CD&R system [1, 22].

2.3.2 Optimised Methods

The optimised method category is the broadest definition of the conflict resolution groups described by Kuchar and Yang. An optimised conflict resolution algorithm is any algorithm that attempts to not only determine a safe path, but also to optimise that path based on a predefined cost function. A cost function is a set of chosen metrics that are used to characterise a path. The selection of these metrics are important since they determined the type of paths that are considered optimal. This is because an optimal resolution method strives to minimise these selected metrics [33]. Therefore the effectiveness of optimised resolution methods depends heavily on the selection of the cost function metrics (see Sections 5.7 for more details on cost function selection).

The broad definition of optimised techniques ensures that multiple methods of determining an optimal path exist. The extensive research done in the field of optimal resolution has resulted in many different approaches to the problem. These approaches include: optimal control theory [22, 34], game theory [35], rule based optimal resolution [36, 37] and genetic algorithms [38]. Of these methods, rule based optimal resolution has attracted extensive research, especially with regard to its implementation through TCAS [29, 39]. Another optimal resolution method that has received attention is combinatorial algorithms. These methods attempt to determine a path using exact algorithms [10]. These algorithms are complete, which means that if a solution exists the algorithm will find it. In order to ensure completeness the algorithms are generally very complicated, making them computationally intractable for complex practical systems. Lastly an effective

approach to optimal resolution that we will be looking into is sampling-based optimal resolution, which has been a popular motion planning method and could be well suited as a practically implemented conflict resolution system.

2.3.2.1 Sampling-based Methods

Sampling-based motion planning algorithms, discussed in depth in Chapter 4, determine a path between two states within a predefined search space. This is accomplished by sampling random points in the search space and checking the viability of the paths to those points with the conflict detection module. These randomly generated paths, determined from the random samples, are used to find a safe conflict free path to a selected goal state. This resolution method is unfortunately not complete, but under certain conditions has been proven to be probabilistically complete [10, 40]. Probabilistic completeness dictates that as the number of samples tends towards infinity, the probability that a safe path will be determined tends towards one if such a path exists. This means that even though sampling-based algorithms are not complete a path will probably be found, if the number of samples selected is high enough. This attribute of a sampling-based algorithm is important, but when applied to a real-time system the rate of convergence of the path probability is a more crucial factor [10, 7]. The two most popular sampling-based algorithms applied to conflict resolution are Probabilistic Roadmaps (PRM) [40] and Rapidly Exploring Random Trees (RRT). These algorithms are explained in depth in Sections 4.5 and 4.6.

In order to implement a real-time system, certain concessions have to be made to ensure that the algorithms are computationally viable. With regard to sampling-based algorithms we concede the completeness of the algorithm for the weaker notion of completeness (probabilistically complete). As the complexity of the scenario increases, so will the number of samples required to ensure probabilistic completeness. The techniques used to sample and determine the path have to be efficient to ensure that a resolution path can be found as fast as possible. This does reduce the guarantee of a solution, but if enough samples are selected then the performance of the system should not be affected.

2.3.3 Prescribed or Rule Based Methods

Prescribed resolution attempts to resolve a conflict scenario by using a predefined set of procedures. These procedures are defined as set resolution manoeuvres that have to be executed according to some set of rules. An example of a purely rule based system is the GPWS: it issues a 'Pull Up' warning when a conflict with the terrain is predicted. This is a set rule and no additional computation is applied to determine a more optimal path. This form of resolution is the most basic form of computerised resolution. Its main advantage lies in that it benefits the pilot, because reactions to prescribed manoeuvres can be trained and performed reflexively [18]. This can reduce reaction time, whereas more complex paths could result in unaccounted-for delayed reactions.

Prescribed resolution is effective in simple scenarios where the assumption is made that no changes will occur to the states of the obstacles, in other words a static environment (like terrain). When the obstacles are dynamic (other aircraft), prescribed manoeuvres are required to be more complicated as the number of

possible scenarios increases. These types of scenarios require additional computation in order to avoid the conflict region and cater for possible changes in the environment. Additionally, prescribed manoeuvres can be inefficient and resolve situations using more aggressive manoeuvres than required [18].

2.3.4 Manual Resolution

Manual resolution is essentially resolution applied by the human pilot using either instrumental flight rules (IFR) or visual flight rules (VFR) to traverse the conflict region. The main advantage of this form of resolution is that it is generally more flexible since it is based solely on human intuition. Additionally the pilot has access to more information than the resolution system. Automated systems attempt to resolve a conflict according to predefined rules that may be unacceptable to the pilot. This means that the manual approach may determine a more optimal path in the eyes of a human [18]. Manual resolution is a viable option in simple scenarios, but a more robust resolution system is required when dealing with complex, cluttered environments. Finally manual resolution is susceptible to human error or misinterpretation of the situation, which can cause significant problems.

2.4 Airport Environment

This section contains a description of what exactly is implied by the term 'airport environment'. We also look into what challenges this environment could pose with respect to a conflict scenario and its possible resolution. In order to understand what protocols have to be adhered to within an aerodrome, we look at the different classifications of the airspace around an airport. The airspace classification determines which ATC regulations apply. Then we look at restricted airspaces and the typical flight profile of a commercial aircraft.

The environment in which a collision occurs has a significant effect on the requirements of the resolution system. The main factor that contributes to the complexity of the airport as an environment is the high volume of air traffic. In 2012 the Airports Council International recorded 930,310 aircraft movements (arrivals and departures) at the Hartsfield-Jackson Atlanta International Airport during that year [41], which calculates to approximately 107 movements an hour. This results in a relatively cluttered dynamic environment when compared to the En Route phase of flight, where cluttered refers to a high density of aircraft in the environment. In addition buildings or complex terrain can increase the complexity of scenario, making an airport environment one of the more challenging resolution scenarios.

2.4.1 Flight rules

Flight rules refer to the rules that apply to an aircraft and pilots while in the sky. The rules applicable to an aircraft depend on (1) the technological capabilities of the aircraft and (2) the pilot's abilities. There are two sets of flight rules: VFR and IFR. VFR generally apply to situations where the aircraft is not equipped with the required instruments or the pilot is incapable of understanding them. This set of rules therefore is subject to more stringent regulations. IFR apply to more advanced aircraft and experienced pilots.

The main requirement for VFR is that the aircraft be flown in conditions of visibility, which means that the aircraft must remain clear of all cloud cover. Additionally a minimum flight visibility of 5 km must be maintained, depending on the classification of the airspace [42]. For a detailed account of the exact VFR requirements under specific conditions see the Air Traffic Management document developed by the ICAO[19].

IFR apply to all aircraft that are equipped with the required navigation equipment. These conditions are less stringent and low visibility flight is permitted. This results in a new set of detailed rules supplied in Rules of the Air by the ICAO as well as in ENR 1.7 - Altimeter Setting Procedures (Section 6) by the Civil Aviation Authority (CAA) [28, 43].

2.4.2 Aerodrome Airspace Classification

Order within an aerodrome is maintained through the definition of different airspace zones. This results in the airspace being classified as either controlled or uncontrolled. Controlled airspace is where the ATC has direct communication and authority over all aircraft within the class. Uncontrolled airspace is where no ATC communications are possible and the aircraft relies on the aid of ACAS.

The classifications of airspace range from A to G. Classes A to E are generally applied to controlled airspaces, while Class G essentially denotes an uncontrolled airspace [44]. Each class adheres to different regulations; these are explained in depth by the ICAO, CAA and the FAA [43, 44, 45]. Table 2.2 explains the different zones and their specific characteristics.

Table 2.2: Airspace Classifications

	A	B	C	D	E	G
Entry Requirements	ATC clearance	ATC clearance	Two-way Communications	Two-way Communications	Two-way Communications	None
Flight Rules Applied	IFR	VFR or IFR	VFR or IFR	VFR or IFR	VFR or IFR	N/A
Airport Requirements	N/A	Radar, Instrument Approaches, High density traffic, Control Tower	Radar, Instrument Approaches, Control Tower	Instrument Approaches, Control Tower	Instrument Approaches	Control Tower
Altitude above Mean Sea Level (MSL)	Above 18 000 ft	From surface to 10 000 ft	From surface to 4 000 ft	From surface to 2 500 ft	From surface to 18 000 ft	N/A

Each of the classes described in the table has different technological capabilities and regulations that govern it. A robust CD&R system should be able to cater for a conflict regardless of the airspace classification. This implies that resolution system should be active in each class, from A to G. Therefore the abilities of a CD&R system should be independent from the controllability of the airspace, making it applicable whether the airspace is controlled or uncontrolled.

2.4.3 Airspace Restrictions

In order to ensure safety on land and in the sky, certain restricted and prohibited airspaces are defined. Different restrictions are placed on aircraft when they enter certain airspaces. These restrictions either prohibit or restrict access to certain airspaces [46]. Typical restricted zones include: Prohibited Areas, Restricted Areas, Danger Areas, Military Training Areas, Areas of Intense Activity, etc. For more information regarding restricted airspaces and their limitations, see the General Rules and Procedures, by the CAA [46]. Restricted airspaces limit the manoeuvrability of an aircraft. This affects the resolution module of a CD&R system by limiting the safe, flyable space. Typically the CD&R system will have access to the location of specific restricted airspaces and favour paths that avoid the restricted zones.

2.4.4 Flight profile

The flight profile of a commercial flight (seen in Figure 2.7) comprises 6 phases. These are: Preflight, Take-off, Departure, En Route, Descent, Approach and Landing. Each of these phases is subject to different conditions and regulations. The preflight and landing phases occur on the runway and all phases apart from the En Route phase occur in an airport environment. This means that a CD&R system should primarily cater for each airborne phase to provide security for the entire flight. Additional features of a CD&R module could cater for runway collisions during the initial and final phases.

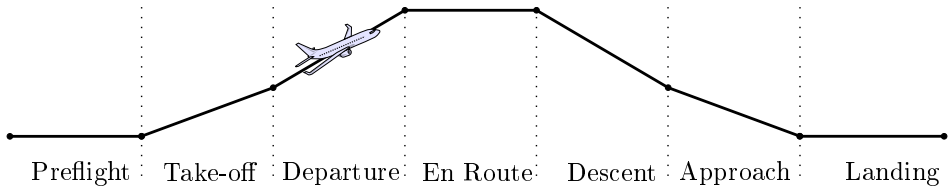


Figure 2.7: Six phases of commercial flight

2.4.5 Challenges

The development of a CD&R system for an aircraft is complicated when applying it to an airport environment in comparison to En Route systems. An En Route CD&R system has to generally cater for single aircraft collisions. This is further simplified by removing terrain avoidance, since the En Route phase of flight is at a relatively high altitude. Therefore applying a system to an airport environment increases the complexity of the problem, due to the vast increase in aircraft density and the incorporation of terrain avoidance. An airport environment represents a cluttered dynamic environment that contains both obstacle aircraft and terrain. The developed resolution system should therefore be capable of dealing with such an environment to ensure its applicability to an aerodrome.

2.5 Summary

The development of a conflict avoidance system for an aircraft is a well researched topic with various methods proposed through the years. The systems currently functioning on commercial aircraft are reliable and detect and resolve conflicts effectively. However, they do possess certain limitations that can be improved upon: TCAS's inability to perform horizontal resolution, the lack of resolution in the EGPWS and the potential for miscommunication between the systems. The research into a system that can reduce these limiting factors has branched into two main groups: force field methods and sampling-based systems. These two groups promote the development of conflict free, dynamically realisable paths in real-time for complex conflict scenarios. The proposed force field methods are attractive, but do possess potential problems. The potential local minima problem is ever present when applying a force field based algorithm, which resulted in these methods not being selected. The development of a conflict resolution system requires that safety and the calculation of a safe path be the primary aims of the system. The probabilistic completeness and fast execution times of sampling-based algorithms resulted in this group being selected for further research.

Chapter 3

Modelling the Aircraft and Environment

The goal of this chapter is to derive simplified models of the aircraft and its environment which are appropriate to be used by the conflict detection and resolution system. The simulation of a designed system is only as accurate as its model of the real world. This section will discuss the methods used to accurately model the required aspects of an aircraft and its environment. Not all aspects of the physical models are essential to the simulation and the simplification of these models is described and justified in the sections below.

We start by defining a model of the aircraft in Section 3.1. Here the general aircraft model is discussed and we propose a simplified model that will be used. Then we explain the modelling of the environment in Section 3.2, defining the differences between static and dynamic obstacles as well as the method used to model them.

3.1 Aircraft Model

The aim of the section is to derive a computationally efficient model that includes all the relevant responses required to successfully model an aircraft. This section is dedicated to the development of a representative aircraft model. The derivation of this model requires the definition of a reference axis system as well as an explanation of the dynamic constraints of a general aircraft. A simplified model for a commercial airliner is then derived that models the aircraft's dynamic response to pilot or autopilot commands given through typical fly-by-wire control systems.

3.1.1 Axis System

In aeronautics there are three different mathematical axis systems used to understand and model the dynamics of a fixed wing aircraft. These are the Inertial, Body and Wind axes. The modelling of the aircraft dynamics requires a predefined reference axis. We wish to model the aircraft as a point mass, and make use of the fly-by-wire commands to control the movement of the aircraft. Since the fly-by-wire commands can be described using the inertial axis system. We only require the inertial axis to model the aircraft, therefore will look at this axis system as well as the relationship between the orientation of the aircraft and this axis system.

3.1.1.1 Inertial or Earth Axis

The inertial axis is an orthogonal system, which can include the North-East-Down coordinate system shown in Figure 3.1. The z-axis (Z_I) points to the centre of the Earth, while the x (X_I) and y (Y_I) axes point North and East, respectively. The inertial axis can be placed at a predefined location of interest, while remaining valid, assuming that the distance travelled is short enough to model the Earth as flat and not rotating [5]. However, this axis system is not really an inertial axis system, but is assumed to be one. The assumption works well for aircraft problems where the aircraft's rotation rate is much larger than the Earth's [11]. Conflicts are generally detected and have to be resolved at close ranges (see Section 2.1.2) therefore the non-rotating Earth assumption can be applied to the system. We will place the centre of axis at the aircraft's current location projected onto the surface of the Earth, when a collision is detected.

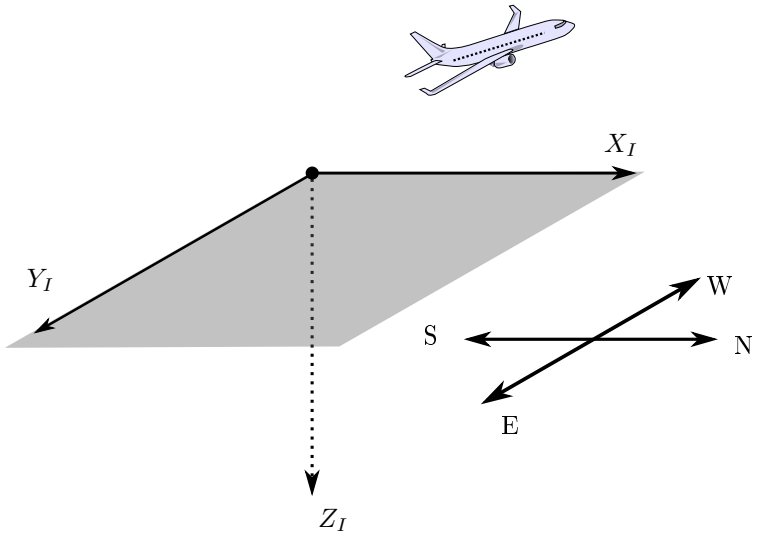
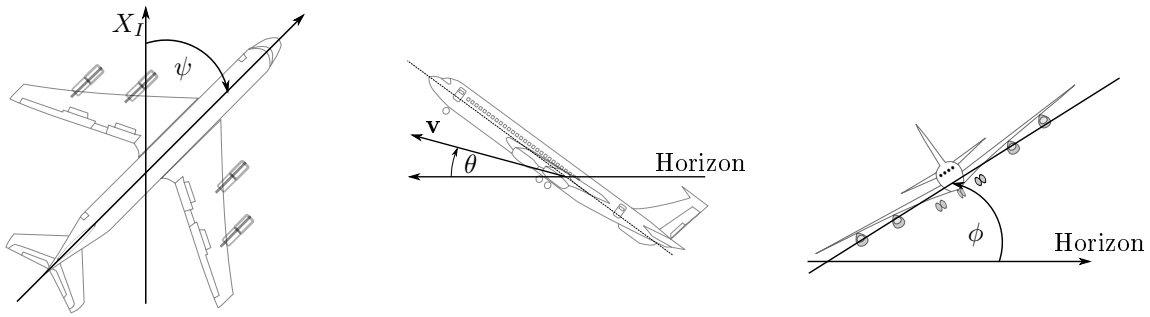


Figure 3.1: The inertial or Earth Axis System for a fixed wing aircraft, placed on the surface of the Earth

The attitude of the aircraft can be is defined as the relationship between the orientation of the body axis system relative to the inertial axis system. This is required when trying to describe the attitude of an aircraft. The body axis system is an orthogonal axis system that is fixed to the aircraft. Its origin is placed at the centre of mass of the aircraft, with the x-axis lying in the plane of summery pointing along some reference line (i.e. the zero angle of attack line of the wing). The y-axis lies perpendicular to the x-axis plane pointing out of the right wing of the aircraft. Finally the z-axis completes the orthogonal axis system, by pointing downward relative to the cockpit [5]. Figure 3.2 shows the relationship between the orientation of the aircraft, velocity vector and the inertial axis system using the angles (ψ, θ, ϕ). Each of the angles is required to govern the control of the aircraft and forms part of the general aircraft model. These angles are defined below.

1. The heading angle, ψ, between the projection of the aircraft's velocity vector and the North axis, is shown in Figure 3.2a, assuming a zero side-slip angle.
2. The flight path angle, θ, is the angle between the velocity vector $\mathbf{v}$ of the aircraft and the horizon as in Figure 3.2b.
3. The roll angle, ϕ defines the angle between the horizon and the wing of the aircraft as seen in Figure 3.2c.



(a) Top view (b) Side view (c) From view

Figure 3.2: Different views of the relationship between the Body and Inertial Axes, adapted from Peddle [5]

3.1.2 General Aircraft Model

The implementation of control systems on commercial aircraft has developed throughout the years. Initially, purely mechanical systems managed the flight control, but due to advances in computer technology digital control methods have arisen [47]. The different control methods that are used progressed as follows: (1) Mechanical Systems, (2) Hydro-Mechanical Systems, (3) Fly-by-Wire Systems and (4) Fly-by-Light Systems. The initial two control methods were purely mechanical, while Fly-by-Wire and -Light are implemented with digital electronics. The development of a digital control system resulted in improved reliability and maintainability of the system as well as some advanced safety control features.

The general Fly-by-Wire system can be divided into four main sub-controllers: Pitch, Yaw, Roll and auto-thrust [48]. These typical controllers are used to manage the aircraft flight path angle, bank angle and velocity vector's magnitude ($\bar{V}$), described in Section 3.1.1. Additionally numerous safety features have been applied to the flight control. These include: bank angle protection, turn compensation, stall and over-speed protection and pitch control and stability augmentation. These protection systems add to the safety and reliability of the controllers. Numerous simulations and flight tests were used to develop and validate the Fly-by-Wire system in general [49]. The control functions present in an aircraft allow for altitude and cross track control. This enables the aircraft to maintain a desired altitude or heading accurately [48]. Additionally the implementation of the autopilot's vertical speed mode allows constant climb or descent control allowing the aircraft to track desired descent or climb rates [50]. The kinetic and kinematic equations of an aircraft can be translated into a 6 degrees of freedom model [5]. This, coupled with a force and moment model, results in a representative model of an aircraft, given that the individual models are accurate. The 6 degrees of freedom model implies that the aircraft can be described using a 6 dimensional state space model: $(x_I, y_I, z_I, \theta, \psi, \phi)$. The state space is comprised of three inertial coordinate parameters (x_I, y_I, z_I) and three attitude parameters (θ, ψ, ϕ).

We assume that a commercial control system implemented on the aircraft will be reliable and accurate, allowing the conflict resolution module to issue commands to the controller. These control inputs will make use of the same commands available to the pilot. The pilot controls the aircraft through the use of the side-stick, pedal and throttle. These devices produce roll and pitch rates, yaw rates and thrust commands.

The conflict resolution module may use these fly-by-wire commands to control the aircraft or may issue instructions to the pilot to execute the fly-by-wire commands manually.

3.1.3 Model Simplification

The digital control systems that manage the flight of a commercial aircraft are generally complex and very intricate. The development of a conflict resolution module requires a definition of the control commands available to the system. The conflict resolution system is capable of either issuing resolution advisories to the human pilot or supplying the on-board autopilot with the relevant reference commands required to executed the desired resolution manoeuvres. This proposed conflict resolution module issues commands to the aircraft autopilot system through the use of different angular rates and a thrust command and it requires a position reference as output from the system. We have simplified the aircraft model to a basic state space implementation with the airspeed command, heading rate command, and climb rate command as inputs and the inertial coordinates of the aircraft trajectory as outputs. The simplified model is controlled using three reference inputs: climb rate ($\dot{h}$), speed ($\bar{V}$) and heading rate $\dot{\psi}$. These commands are used to control the position of the aircraft in space. The bank angle command is used to determine the required reference heading rate. Additionally slew rates and time delays are added to the design to model possible delays due to the controller. A representation of the control design is displayed in Figure 3.3. The saturation blocks ensure that the command limits of the aircraft are not exceeded, while the transient responses, represented by low-pass filters (LPF), model the transition time required to reach a given reference command. Additional potential delays in time are modelled using the time delay blocks e^{-st_d}.

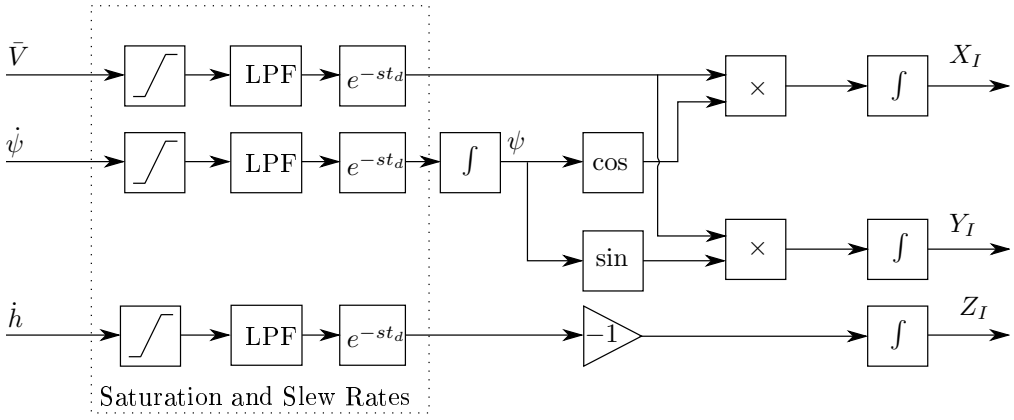


Figure 3.3: Block diagram of the simplified aircraft model

This model can be further simplified by removing the transient response and time delay blocks. Due to the nature of the simulations and the distances simulated we assume that the effect of the time delays and slew rates are negligible when compared to the entire simulation times.

3.1.4 Differential Constraints of an Aircraft

The differential constraints of the aircraft refer to its physical or dynamic limitations. The conflict resolution module has to be aware of the limitations with regard to the control inputs in order to ensure that the

calculated paths are indeed feasible. Therefore knowledge of the different physical limitations of an aircraft is required in order to select control inputs that can realistically be followed.

A commercial aircraft has certain limitations on the control inputs due to many different factors, including aerodynamic, structural and technological constraints [51]. The aircraft speed is bounded by a constraint on propulsion, due to the limitations of the engines, structural constraints and a stall constraint, due to the aerodynamics of the aircraft. Bank angle commands are also limited due to multiple factors: lift coefficients, velocity and structural bounds. The maximum speed of an aircraft is measured by a certain Mach number. Mach refers to the ratio between the aircraft speed and the speed of sound in the same surroundings. Therefore the Mach number translates to a specific speed at a certain altitude, that changes based on the air density. This means that the speed of an aircraft is not limited based on a predefined speed, but instead on a varying speed based on the altitude. The speed limitations are therefore dependent on the aircraft's assigned Mach number and the current altitude [44]. The standard rate of a turn is defined by the FAA as 3° per second [44]. In order to maintain this standard rate of turn the reference bank angle is a function of the speed of the aircraft. This means at higher velocities a lower reference bank angle is commanded. The standard bank angle is assumed to be 18°, according to the United States Standard for Area Navigation, with a maximum angle of 25°, and 3° below $500ft$ near airports [52].

Different manoeuvres result in an increased aircraft load factor. The load factor is defined as the amount of g-force exerted on the aircraft. These forces are caused by stress placed on the aircraft when it deviates from straight and level flight [13]. The current load factor is important because it is possible to overload the aircraft structures and because an increase in the load factor increases the stalling speed of the aircraft. This implies that, in order to ensure that a path determined is flyable, we have to ensure that the derived path does not impose a large load factor on the aircraft. Aircraft are designed to withstand a certain maximum load factor limit, as shown in Table 3.1. The determined path must ensure that the load factor limitations will remain within the bounds depicted in the table when traversed.

Table 3.1: Load factor limitations for different aircraft catagories, adapted from the FAA [13]

Aircraft Catagory	Load Factor Limit
Normal*	3.8 → -1.52
Utility**	4.4 → -1.76
Acrobatic	6.0 → -3.00

* A safety factor of 50 % is given to aircraft with a gross weight larger than 1814 kg.
** Includes aircraft performing mild acrobatics, as well as spins.

The load factor can be determined from the specific climb or bank angle and the current aircraft velocity described by

$$\eta = \frac{\mathrm{L}}{\mathrm{W}}, \tag{3.1}$$

where η depicts the load factor and L, W represents the aircraft lift and weight. Equation 3.1 allows the calculation of the current load factor of the aircraft based on the required lift and weight for each input command. Using this, in conjunction with Table 3.1, it is possible to determine the differential limitations of the aircraft for all required paths at its current speed.

If the limitations of the aircraft are exceeded, certain upset conditions can occur. A few examples of possible upsets are described by Wainwright et al. [53]. Upset conditions include climb and descend angles that exceed 25° and 10° respectively, as well as bank angles of 45°. These inputs can be used as guidelines for the selection of possible bank angle and climb rate limitations. The FAA limits the maximum bank angle command for an average general aviation aircraft at 60° [13]. The differential constraints of different aircraft vary dramatically, especially when considering military and acrobatic planes. The selection of the control inputs can be based on many different factors. We have based them on the limitations and bounds explained above as well as the resolution commands used in the existing resolution systems (see Section 2.2).

3.2 Environment Model

The environment model describes the algorithm's interpretation of the environment. The selection of the model does not affect the conflict resolution module, but has a large impact on the detection module's computation time and efficiency. An efficient and effective method of modelling the environment is required in order to apply a basic conflict detection module. The modelling of an aircraft's conflict region as well as the conflict regions of all other aircraft in the environment can be very costly. This section discusses methods used to model both the static and dynamic obstacles in the environment as well as the host aircraft.

3.2.1 Static and Dynamic Obstacles

Conflict resolution requires the generation of a conflict-free path that avoids the conflict region. This conflict region can be populated by either static obstacles (terrain and buildings) or dynamic obstacles (aircraft). A successful conflict resolution module has to determine a path that avoids both these types of obstacles. Static obstacles are easier to cater for as their locations in the environment remain constant. Dynamic obstacles, however, cause some complications. In a dynamic environment the time at each position on the path becomes important. In order to test for conflict, the algorithm has to check the location of the host and obstacles at each time along the predicted path. Adding time as a dimension increases the computational complexity of the algorithm, and therefore an efficient and effective method is required to cater for the time domain [7]. Catering for dynamic obstacles adds time as a state to the conflict detection module. Different methods used to incorporate time as a state are discussed in Sections 4.3 and 4.6.

The conflict resolution module we propose in this thesis will cater for both dynamic and static environments and will therefore require a complex enough conflict detection module. The conflict detection module needs access to an environmental model that can be easily and effectively implemented (see Section 3.2.2).

3.2.2 Minkowski Addition

Minkowski developed a mathematical method that enables the modelling of the host aircraft as a point in space. This removes the complexity of modelling the host's body by incorporating the configuration or state of the host as part of the obstacles, where the configuration of an obstacle includes all possible translations

and rotations of the obstacle in the environment. This can be accomplished through the use of Minkowski addition, defined below [54, 55].

Lets denote an arbitrary host vehicle as a set of points $\mathcal{H}(\vec{p})$, where its location is defined by an arbitrary translation $\vec{p} = (x, y)$. Given an obstacle $\mathcal{O}$, the conflict region (Ω_c) is defined, in Equation 3.2, as all configurations where $\mathcal{H}$ intersects $\mathcal{O}$. We assume that the configurations of $\mathcal{H}$ define the conflict region of the host vehicle.

$$\Omega_c = \{\vec{p} \mid \mathcal{H}(\vec{p}) \cap \mathcal{O} \neq \emptyset\} \tag{3.2}$$

A visual representation of Equation 3.2 is that the edge of the conflict region is traced by the reference point of the host as it slides around the edge of the obstacle. Minkowski addition is a method that can be applied to determine this region. The Minkowski addition of two sets S_1 and S_2 is defined as the pairwise addition of the points in each set, i.e. the convolution, as seen in Equation 3.3. Equation 3.4 states that a negative set is defined as the inverse of all the points in the set.

$$S_1 \oplus S_2 = \{\vec{p}_1 + \vec{p}_2 \mid \vec{p}_1 \in S_1, \vec{p}_2 \in S_2\} \tag{3.3}$$

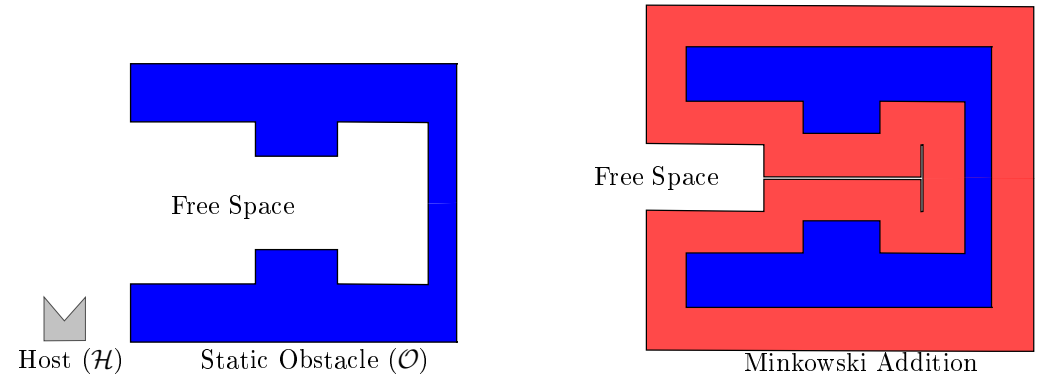
$$-S = \{-\vec{p}_1 \mid \vec{p}_1 \in S\} \tag{3.4}$$

From the Equations 3.3 and 3.4 we can define the conflict region as the Minkowski addition of the obstacle and the inverse configuration of the host:

$$\Omega_c = \mathcal{O} \oplus (-\mathcal{H}). \tag{3.5}$$

The Minkowski addition of the host and obstacles results in a redefinition of the free space in the environment as seen in Figure 3.4. In Figure 3.4b the host is modelled as a point mass, while the configuration of the obstacle is adapted to include the Ω_c region, in red.

Minkowski addition is a very effective method of simplifying the environment by modelling the host as a point mass. This simplification reduces the computational complexity required to perform conflict detection. A drawback of the general Minkowski addition is that for each different configuration of $\mathcal{H}$ a new Ω_c region of the obstacles has to be calculated. In order to reduce these additional calculations we model all the aircraft in the environment as spheres. This greatly simplifies the Minkowski addition, because every orientation of a spherical body results in an equivalent configuration, therefore the Minkowski addition has to be determined only once for the environment. Additionally, calculating the Minkowski addition using spheres is very simple. If we assume that $\mathcal{H}$ and $\mathcal{O}$ are both spherical aircraft, the calculated Ω_c region is a sphere centred at $\mathcal{O}$ with the radius $r_{\Omega_c} = r_{\mathcal{H}} + r_{\mathcal{O}}$, where r denotes the spherical protected zone radius. This favourable characteristic of a spherical host conflict region enables the modelling of the terrain and other obstacles as their original configurations with an offset of $r_{\mathcal{H}}$. This greatly simplifies the modelling of both the obstacles and the terrain as well as reduces the computational complexities required to model them.



(a) Configuration of host and static obstacle (blue)

(b) Configuration of the environment after Minkowski Addition, the addition of the host's configuration to the obstacle's is shown in red

Figure 3.4: Representation of the change in configurations due to Minkowski addition, adapted from Hachenberger [6]

3.2.3 Uncertainty

Uncertainty is a crucial aspect of CD&R. We have already discussed the three methods used in conflict detection to model the predicted trajectories of the aircraft in the environment in Section 2.1.2. Van Daalen [7] highlights two path planning methods used to cater for uncertainty in the environment by Hsu et al. [56] and Fulgenzi et al. [57]. The first method proposed by Hsu et al. projects the obstacle along its current trajectory forward in time, while increasing the conflict region of the obstacle as the prediction time increases. Additionally Hsu et al. also continuously re-plans at fixed time intervals. The method proposed by Hsu et al. corresponds to the worst case propagation technique, making this method prone to false alarms and unnecessary replanning. Fulgenzi et al. developed a method that models the static environment as a probabilistic occupancy grid[1], while modelling dynamic obstacles using a Gaussian process. The total probability is determined by summing the probability of conflict along each path segment. This method produces a lower bound on the probability of conflict, meaning that it underestimates it, resulting in the possible generation of potentially unsafe paths.

A method proposed by van Daalen [7], illustrated in Figure 3.5, simplifies the development of the resolution system by shifting the responsibility of uncertainty management to the conflict detection module. By making use of a probabilistic conflict detector we can entirely remove uncertainty from the resolution module. This is possible because the resolution module is separated from the environment by the detection module. Therefore the detection module is tasked with managing uncertainty, allowing the application of a general resolution system to these uncertain environments.

We have simplified the system described above by assuming a certain environment, due to the fact that the probabilistic conflict detection module is being developed in parallel by Pienaar [16]. The conflict resolution module will make use of the system shown in Figure 3.6. This structure is advantageous because the only difference between the certain and uncertain systems are the detection modules. As seen in the figures the

[1] An occupancy grid defines the environment as a lattice of cells, where each cell contains the probability that it is occupied [58]

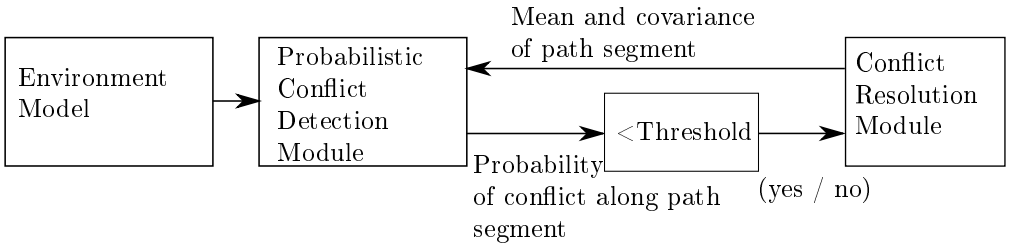


Figure 3.5: Seperation of Resolution module from an uncertain environment by the Detection module, adapted from van Daalen [7]

conflict resolution module makes use of the same inputs and outputs, regardless of the detection module and environment. This enables the application of the same resolution system to a certain environment, while uncertain environments can be handled by replacing only the detection module. Therefore the resolution system used can be tested on a certain environment and should theoretically provide similar results in an uncertain environment given an effective probabilistic conflict detector and a realistic threshold selection.

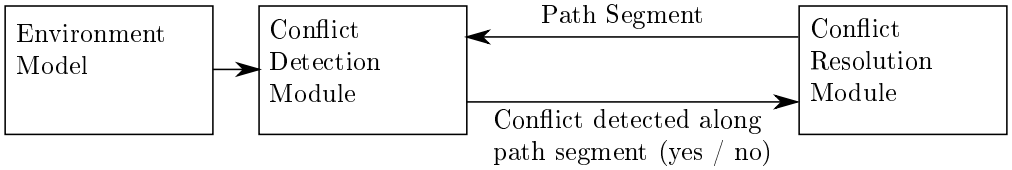


Figure 3.6: Seperation of Resolution module from the environment by the Detection module (No Uncertainty), adapted from van Daalen [7]

3.3 Holonomic versus Non-Holonomic Vehicles

Holonomic systems are systems that are only kinematically constrained by

$$f(s_1, s_2, \ldots, s_n, t) = 0 \tag{3.6}$$

while non-holonomic systems are constrained by

$$f(s_1, s_2, \ldots, s_n, \dot{s}_1, \dot{s}_2, \ldots, \dot{s}_n, t) = 0, \tag{3.7}$$

where f denotes a function, s_i represents the position states and $\dot{s}_i$ the velocity states of a vehicle [59]. Systems that do not adhere to either of these constraints are generally grouped under the non-holonomic category.

A holonomic system is one where the constraints on the system can be reduced to Equation 3.6. This holonomic constraint translates to a constraint on the configuration of the system, defining the states that cannot be reached [60]. A non-holonomic system cannot be reduced to Equation 3.6, and is thereby constrained by velocity, but has no constraint on its position states. This means that a non-holonomic system can reach any configuration or state, but has to conform to a set of velocity constraints in order to reach it. Assume that

the motion of a vehicle is defined by a certain curve. A holonomic vehicle has no velocity constraints and can therefore be represented by any curve, while a non-holonomic vehicle can only move along certain curves that are compatible with the system's velocity constraints [61]. A non-holonomic vehicle can therefore not move in any arbitrary direction at a given time instant, making the vehicle path-dependent. Path-dependence implies that the current state of the vehicle is determined by the path taken to reach it. Simply stated a non-holonomic system contains some states that cannot be instantaneously changed, but could be reached by a controller through the execution of certain manoeuvres.

3.4 Summary

The development of a conflict avoidance system required the modelling of the entire environment, including all aircraft (host and obstacle) and terrain. The application of an accurate conflict detection module requires an accurate representation of the environment. The applied system makes use of Minkowski addition to simplify the modelling of the aircraft as well as the environment. Minkowski addition in conjunction with a spherical conflict region assumption allows the conflict detection module to model the host aircraft as a point in space and the terrain and all obstacle aircraft as their conflict regions with an offset (see Section 3.2.2 for details). The application of a resolution system that derives differentially realisable paths requires an accurate model of the host aircraft. The development of a simplified aircraft model on which the system is applied can be seen in Section 3.1.2. The identification and selection of a set of maximum constraints are required to ensure that the paths calculated by the resolution system are indeed flyable. The identification and selection of these constraints are discussed and justified in Section 3.1.4. This chapter highlights the methods required to accurately model both the aircraft and terrain to ensure that the paths calculated are flyable and that the simplified conflict detection module that is applied, functions efficiently and accurately.

Chapter 4

Sampling-Based Algorithms

The aim of this section is to discuss the attributes of the different sampling-based path planning algorithms and to determine which are most appropriate for implementing a conflict resolution system for commercial aircraft in airport environments. Sampling-based algorithms have been applied to a large range of different scenarios, which include solving puzzles, automotive machinery, moving furniture, developing intelligent computer game characters, artificial intelligence and motion planning. Each scenario has different goals, but the basic algorithms are similar. The algorithms attempt to create different paths to traverse the conflict region by connecting randomly sampled points [62]. Every system has a state, an execution time, a set of possible actions, an initial and final state, a set of criteria and a required plan. These aspects of the system are described below.

1. The state of the object is its current position, shape and orientation. Changes can be made to certain states (position, orientation), while others have to remain constant (shape). This is dependent on the objects in question.
2. The execution time can be defined as the time taken to execute the plan.
3. The possible actions are determined by multiple factors including but not limited to the object's parameters, weight and size. Only applicable actions can be used to move it. These have to be adhered to when determining the path, otherwise it can not be executed.
4. The initial state is the position and orientation of the object at the beginning, while the final state is the desired or goal state that solves the problem.
5. The criteria define how the object should be moved. It is generally based on either feasibility or optimality. Feasibility infers that a path is found regardless of its efficiency. Optimality attempts to determine paths that optimise certain performance criteria. This can be the shortest path, fastest path, least amount of energy used, etc.
6. Finally the plan is how the object will be moved from its initial state to the final state, while avoiding all obstacles.

A problem can be considered solved if the plan dictates a path that moves the object from the initial state to the final state within the execution time, while adhering to the possible actions and criteria. The generalised mover's problem [63, 10, 9] is a very basic case of a sampling-based motion planner. It requires a single plan to traverse the space and is generally not limited by the actions that can be taken. More complex examples contain dynamic obstacles, uncertainty and dynamic constraints, which limit the available actions that can be taken.

The sampling-based algorithm's most attractive feature is its fast execution time. This makes it possible to determine a path in real-time or near real-time. However the algorithms lack of completeness is a negating factor. Granted if enough samples are used and the convergence rate is high enough, then the algorithm applied is a viable resolution system. The field of sampling-based path planning has yielded numerous effective algorithms. Of these, the Probabilistic Roadmap (PRM) and Rapidly-exploring Random Trees (RRT) are amongst the most influential [10, 62]. These algorithms are explained in depth in Sections 4.5 and 4.6, respectively.

The chapter starts by looking at the different concepts required to understand the terminology used with regard to sampling-based algorithms in Section 4.1. This is followed by a brief explanation of the general functionality of a sampling-based algorithm in Section 4.2, as well as descriptions of the components required to implement a sampling-based path planner in Sections 4.3 and 4.4. We conclude the chapter by reviewing three prominent sampling-based techniques in Sections 4.5, 4.6 and 4.7.

4.1 Concepts and Notation

In order to fully understand the functionality and methodology behind sampling-based algorithms, there are a few key notational concepts that have to be made clear. Here follows a brief description of these concepts as well as the notation used in this chapter.

- All path planning and resolution methods discussed here will be applied to a certain host, denoted by $\mathcal{H}$. This host is an arbitrary vehicle that will have to traverse the space by following the path determined by a planner.
- The configuration space, described in Section 4.1.1, is the environment in which the path planner has to determine the path. It is known as the C-space and is denoted by $\mathcal{C}$.
- The state of the host is a representation of all the relevant information regarding the host within the C-space. For the purposes of high-level path planning, an aircraft can be described by a minimum of 6 states $[x, y, z, \theta, \psi, \phi]$ (see Section 3.1 for details). The states of the aircraft are defined relative to the axis describing the C-space of the system. The initial state q_{init} describes the aircraft state at the starting point, while the goal state q_{goal} is the state that the planner aims to reach. These two states are collectively known as the boundary states.
- All data determined by a sampling-based algorithm has to be stored in some structure. Generally a type of graph G is used to describe the C-space (see Section 4.1.2). Each graph is made up of a set of vertices and edges $G = (V, E)$. A subset of the general graph structure is a tree T (see Section 4.1.2 for a description).

- The set of edges E and vertices V within a graph describe the C-space as a set of states and paths between them. Each vertex will contain a reachable state, where a single vertex is denoted as $v \in V$. A single edge, $e \in E$, describes a path between two vertices, where $e = (v, v')$ is an edge from v to v'.
- A path τ, described in Section 4.1.3, represents all states that make up an edge.
- Sampling-based planning takes random samples of the C-space, see Section 4.3, where the sampled state is denoted by n. Generally a sampling-based planner has a set maximum sample number n_S at which the algorithm terminates.
- In order to determine a viable resolution path, a method is required to determine if the path is indeed safe. This is accomplished through Conflict Detection (see Sections 2.1.2 and 4.4 for details on different methods of detection).

4.1.1 Configuration Space

Choset et al. state that one major complication that arises when planning a path is to ensure that every point of the host is conflict-free. In order to do this we have to map each point of the entire host [9]. The configuration space or state space ($\mathcal{C}$) is an answer to this problem. It represents the space that all the possible transforms or configurations of the host can exist in, thereby encompassing any state that the host can reach [10, 9]. This allows the path planner to model the host as a point in the configuration space or C-space, denoted by $\mathcal{H}$ [63]. The simplification is achieved through the use of the Minkowski addition (see Section 3.2.2). In order to successfully model the host, the C-space requires the same number of states as the host's degrees of freedom. This means that in order to model an aircraft's configuration space we need $\mathcal{C} \subseteq \mathbb{R}^6$. This is because there are 6 states required to fully model the three dimensional environment in which an aircraft exists, namely $[x, y, z, \theta, \psi, \phi]$ (see Section 3.1).

The aim of all path planning algorithms is to capture the connectivity of the modelled C-space. Any space can be considered connected if it cannot be represented by the union of two non-empty, disjoint open sets [63]. An even stricter definition of connectivity is whether the space is path connected. The C-space can be considered path connected if any two states can be linked with a continuous path. The path planning algorithms described here will assume that the modelled C-spaces are both connected and path connected, and simply refer to both conditions as the connectivity of the space. The path planning algorithms attempt to capture the connectivity of the space by creating numerous vertices and edges in order to model the C-space as a graph of path connected states.

4.1.2 Graph Theory

In order for a sampling-based algorithm to capture the connectivity of a space and successfully determine a viable resolution path, the data describing the C-space has to be stored [8]. The C-space is connected using multiple sampled states or vertices (V) connected through edges (E). A graph ($G = (V, E)$) is a collection of these vertices and edges [9]. Each vertex in a graph typically refers to a specific state in the C-space that the host can reach. These states represent all the information required to model the host in the C-space (see

Section 4.1.1). For an aircraft all 6 of its states are stored. The edges of a graph denote the relationship between certain vertices. This relationship can be defined in different ways. Generally, in motion planning, this metric is defined as a viable path τ, described in Section 4.1.3. The definition of the edges determines whether a graph is directed or undirected. If the path described by an edge is directed from one vertex to another then the graph is directed. If no direction is specified then the graph becomes undirected. Figure 4.1 depicts a general undirected graph containing vertices and edges.

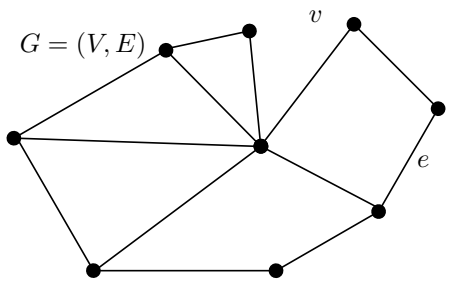


Figure 4.1: Basic undirected graph $G = (V, E)$.

The connectivity of a graph is determined by the relationship between the vertices and the edges. A graph is considered connected if all the vertices can be linked through some path τ, described in Section 4.1.3. This means that each vertex in the graph is linked to some other vertex through an edge [8].

4.1.2.1 Tree Structure

In certain path planning problems we require the use of a specifically structured graph. One of these types of graphs is a tree T, where $T \subset G$. A tree structure is a special graph that contains no cycles (seen in Section 4.1.3) and is connected. This means that any two vertices of T are linked by a unique path, as seen in Figure 4.2 [8]. Certain tree structures, known as rooted trees, contain a root from which the tree stems. Generally, in path planning, rooted trees are used, where the root denotes the initial or goal state.

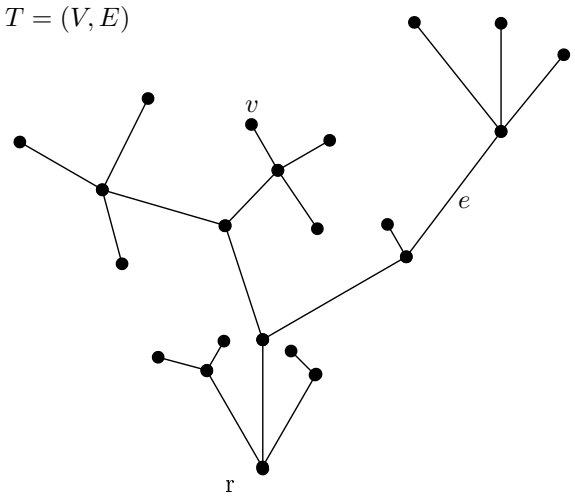
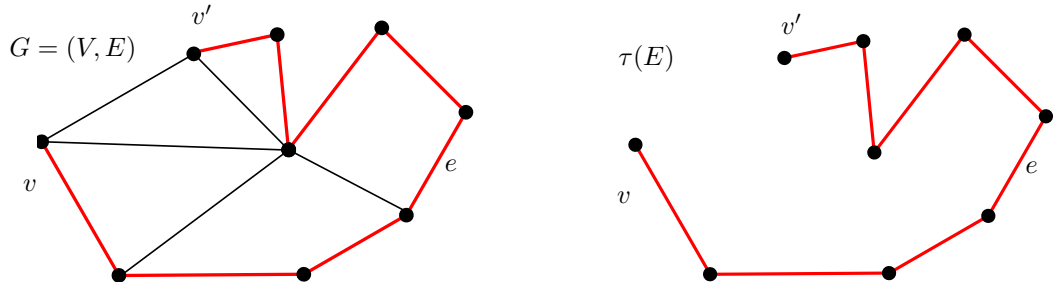


Figure 4.2: Basic undirected tree T stemming from root r, adapted from Diestel [8]

4.1.3 Path

One key aspect required to implement a successful path planner is the ability to determine whether a connected path exists between two points in space. What this means is that the planner is connecting two points in the configuration space using a continuous path [10]. Figure 4.3 shows a typical path from vertex v to v' determined within a graph structure. Note that a graph is an abstract representation of the search space meaning there is no connection between an edge in a graph and the path represented by that edge; the edge simply shows that the two vertices are or can be connected.

Let the configuration space be represented by $\mathcal{C} \subseteq \mathbb{R}$, where a path is defined as $\tau : [0, 1] \rightarrow \mathcal{C}$. This path defines a function, where each point along the path is denoted by $s \in [0, 1]$. Therefore $\tau(s)$ represents each point along the path. A completed path from the initial state to the goal state (q_{init} to q_{goal}) would imply that $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$, given that no cycles occur [63, 10]. A cycle is where the same vertex is visited more than once during a path [8]. This is an undesirable path trait with regard to motion planning and should be avoided.



(a) Path identified in graph $G = (V, E)$ (b) Extracted Path $\tau(E)$

Figure 4.3: Extracted path τ from graph G, adapted from Diestel [8]

The practical application of the path determined by the sampling-based algorithm is generally implemented as a set of manoeuvres. This application is controlled by a Local Planning Method (LPM) (see Section 5.6 for more details). The edges shown in Figure 4.3b may not result in practically viable paths and generally complex LPM functions are used to ensure that the differential constraints of the host are adhered to.

4.2 Overview of Sampling-Based Planning Algorithm

The aim of a sampling-based algorithm is to model the connectivity of the C-space accurately and effectively. Sampling-based algorithms therefore attempt to replicate this characteristic through the use of graphs and different sampling methods. Sampling-based path planning makes use of different sampling methods to populate these search graphs $G(V, E)$. The different methods of sampling the space, as well as modelling the sampling region, are discussed in Section 4.3. The composition of the search graph is determined by how the edges and vertices are linked together. There are two main data storage graph structures used in path planning; these are: (1) graph and (2) tree structures (see Section 4.1.2). The functioning of a general basic sampling-based algorithm is described below.

1. Initialisation: during this phase a search graph $G(V, E)$ is created and populated with the boundary states. The set of edges is initialised with an empty set ($E = \emptyset$) and the set of vertices is set to the initial or starting state ($V = q_{init}$).
2. The Graph Population phase is where the path planner uses some form of sampling strategy to populate the vertices of the graph. $V = n_i$ given that $i \in [0, n_S]$, where n_i is the i'th sample and n_S is the predefined total number of samples desired. All the vertices are connected by edges.
3. Graph Searching is used to determine the shortest path from the initial to the final state. The different methods of searching through the graph are discussed in Appendix A.
4. When the shortest path has been determined it is checked for conflict using a conflict detection module, described in Sections 2.1.2 and 4.4.
5. If a conflict-free path has been found it is returned as the planned path of the system; if not, the algorithm returns to Step 2.

The different methods of sampling-based resolution are distinguished by how each of the steps above is executed. Each alternative execution method has different benefits, but most general sampling-based algorithms follow the steps described above.

4.3 Sampling Strategy

The overall performance of all sampling-based planners are strongly linked to how effectively the connectivity of the C-space can be captured. Ideally we aim to sample each state of the C-space. In order to achieve this the C-space has to be modelled as a set of finite sampled states. A complication arises here because all C-spaces are uncountably infinite, therefore it is practically impossible to capture every state [10].

A prerequisite for path planning is that the sampler produces samples that are dense with a probability of one. A set U can be considered dense in Q, if the closure of U equals Q (cl(U) =U). This practically implies that if we look at a random non-zero sized section of the C-space, then as the number of samples tend towards infinity the probability that a sample falls within this section tends towards one. There always exists a trade-off between effectively capturing the connectivity of the space and computational complexity. When dealing with a cluttered environment a higher number of samples is generally required to successfully capture its connectivity. Therefore it becomes especially important to determine the optimal number of samples required to ensure that a viable graph is constructed with minimal computation. There are many different sampling techniques that adhere to this requirement, explained by LaValle [10] and Choset et al. [9]. Of the numerous sampling techniques the most general method is uniform sampling, additionally it is generally required as part of the probabilistic completeness proof.

4.3.1 Uniform Random Sampling

A sampling strategy can be categorised as uniform when the probability density function (PDF) of the samples over the C-space is uniform [10]. Uniform sampling methods are dense with a probability of one.

This attribute makes them viable as a sampling strategy for path planning. Uniform sampling is one of the simpler methods of sampling with regard to implementation. Theoretically a malicious obstacle that attempts to ensure a conflict can not beat a planner that is applying this random sampling method [9]. When applied to complex environments it has been noted that the computation time of uniform samplers tend to vary dramatically and when the sampler has to find a narrow passage it tends to fail. Regardless of these disadvantages, in most practical cases with high degrees of freedom, uniform sampling strategies work effectively [9]. Additionally the narrow passage problem will not, for all practical purposes, affect aircraft applications.

4.3.2 Sampling in Time

Sampled states generally contain all the information required to model the host in the C-space (see Section 4.1.1). It can be advantageous to not only sample the host's potential states, but also a random arrival time at which the host will reach this state. This means that the host will have to adjust its speed based on the sampled vertices' arrival times. The viability of the arrival times can be checked to ensure that they are dynamically reachable. Sampling in time increases the dimension of the C-space, but vastly simplifies conflict detection within dynamic environments.

4.4 Conflict Detector

Ensuring that the states sampled are free of conflict is of paramount importance, especially in a CD&R system. We achieve this through the use of a conflict detection module. Generally the majority of the computation time of a CD&R system is spent detecting conflicts [64]. This section describes the basic aspects of a conflict detector, but for all theoretical purposes we model it as a black box [10]. There are many different methods of conflict detection, Section 2.1.2 gives an overview of the general differences between detection modules and methods of state propagation. The aim of the conflict detector is to predict whether a conflict will occur in the future [18].

Conflict detection in a static environment is generally a simple task in comparison to dynamic environments. In a static environment time is not a factor when detecting conflict, since all potential conflict's locations are constant, whereas in a dynamic environment, due to moving obstacles, time plays an important role in the detection process. In order to accurately determine whether a path is indeed conflict-free, the arrival time and flight time of each vertex and edge are required. Either this has to be calculated based on velocities and acceleration or through sampling in time. When sampling in time a vertex can be tested for conflict before a path has been determined from the initial state. This helps when attempting to create a conflict-free search graph before searching for a path. Different conflict detection algorithms have been determined that range from the purely theoretical, computationally complex methods to practical, performance based systems. Kuchar and Yang reviewed numerous methods of conflict detection [2, 18].

Generally the resolution and detection modules or systems are developed separately and can be integrated once completed. The method of conflict detection does not affect the resolution module due to the system structure described in Section 3.2.3, therefore different conflict detection strategies can be applied to the same resolution system.

4.5 Probabilistic Roadmap

The term Probabilistic Roadmap (PRM) was first coined in 1996, by Kavraki et al. in their paper "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces" [40]. This method of path planning stemmed from the research of Overmars in 1992 [65]. Since then, extensive research has been done on the topic [66, 67, 68, 69]. The basic PRM algorithm explained in this section is derived from the algorithm described by Kavraki et al. [40] and Latombe [63], which has been proven to be probabilistically complete. This path planner is feasible in high-dimensional problems and can work efficiently for up to 12 degrees of freedom.

4.5.1 Basic Algorithm

The PRM planner divides the path planning problem into two separate phases: the learning phase and the query phase. During the learning phase the algorithm creates a randomly generated roadmap, then the query phase adds the initial and final states (q_{init}, q_{final}) and determines the path to goal. The roadmap generated in the learning phase is represented by an undirected graph G, as in Section 4.1.2. This graph is made up of vertices and edges $G = (V, E)$.

Learning Phase

The learning phase, described in Algorithm 1, determines the search graph or roadmap by randomly sampling vertices within the predefined sampling space (C-space). The sampling process continues to sample random vertices until the desired number of conflict-free vertices (n_S) have been added to the graph. Note that in a dynamic environment only vertices that have been sampled in time can be checked for conflict.

Once this is complete the algorithm generally makes use of a k-nearest neighbour search to determine which vertices are connected, where k is the maximum number of edges connected to each vertex. The nearest neighbour search can define a nearest neighbour using any measurement metric; generally Euclidean distance is used [70]. Each vertex is connected to its k-nearest neighbours K_n through paths determined by a local planner (LPM) and, if conflict-free, is stored as an edge. Once all the edges of each vertex have been added to the graph, the construction of the roadmap is complete. The aim of the roadmap is to successfully capture the connectivity of the C-space. If the initial roadmap fails to achieve this, adaptations can be made to it [9]. One example is adding extra vertices if a path can not be found.

Algorithm 1 PRM - Learning Phase, adapted from Choset et al.[9]

1: $V \leftarrow \emptyset$
2: $E \leftarrow \emptyset$
3: **while** $|V| < n_S$ **do**
4: **repeat**
5: $n \leftarrow$ randomly sampled state in $\mathcal{C}$
6: **until** n is conflict-free
7: $V \leftarrow V \cup n$
8: **for every** $n \in V$ **do**
9: $K_n \leftarrow$ k-nearest neighbours of n, where $K_n \subset V$
10: **for every** $k_n \in K_n$ **do**
11: **if** $(k_n, n) \notin E$ and $\tau(k_n, n) \neq \emptyset$ and $\tau(k_n, n)$ is conflict-free **then**
12: $E \leftarrow E \cup (k_n, n)$

Algorithm 1 contains a few functions that may appear unclear to the reader. For clarity please refer to Sections 4.3 and 5.5 on sampling strategies for a description of line 5. The methods used for determining a conflict, described in line 6 and 11, are discussed in Section 4.3 and the LPM function discussed in Section 5.6 describes the calculation of the path τ in Line 11.

Query Phase

Once the learning phase is complete, we have a successfully populated roadmap that captures the connectivity of the C-space [9]. Now during the query phase, depicted in Algorithm 2, the focus of the algorithm shifts to determining a viable path from the initial state to the goal state. Before this can be accomplished, these two boundary states have to be added to the roadmap. The conflict-free edges of the states are determined using the k-nearest neighbour search, and a local planner to find their corresponding paths. All that remains is determining the optimal safe path, by making use of any of the search algorithms described in Appendix A. If a successful path has been determined, the path P is returned as the conflict resolution path.

Algorithm 2 PRM - Query Phase, adapted from Choset et al.[9]

Input
$q_{init} \rightarrow$ the initial state
$q_{goal} \rightarrow$ the goal or final state desired

1: $V \leftarrow V \cup \{q_{init}\} \cup \{q_{goal}\}$
2: $K_{q_{init}} \leftarrow$ k-nearest neighbours of q_{init}, where $K_{q_{init}} \subset V$
3: $K_{q_{goal}} \leftarrow$ k-nearest neighbours of q_{goal}, where $K_{q_{goal}} \subset V$
4: **for every** $q' \in K_{q_{init}}$ **do**
5: **if** $(q_{init}, q') \notin E$ and $\tau(q_{init}, q') \neq \emptyset$ and $\tau(q_{init}, q')$ is conflict-free **then**
6: $E \leftarrow E \cup \{(q_{init}, q')\}$
7: **for every** $q' \in K_{q_{goal}}$ **do**
8: **if** $(q_{goal}, q') \notin E$ and $\tau(q_{goal}, q') \neq \emptyset$ and $\tau(q_{init}, q')$ is conflict-free **then**
9: $E \leftarrow E \cup \{(q_{goal}, q')\}$
10: $P \leftarrow$ shortest path from q_{init} to q_{goal}
11: **if** $P \neq \emptyset$ **then**
12: **return** P
13: **else**
14: **return** failure

4.5.2 Discussion of the Basic PRM

Generally the Learning phase of the basic PRM described above is computed off-line, meaning that the roadmap is created before the host starts moving. Then when a collision is predicted the query phase adds the host's current position as the initial state and attempts to determine a safe path to the goal. This implies that generally one roadmap is determined, but multiple queries can be made, from different locations within the C-space. This is commonly know as a multi-query planner [9]. This form of path planning is effective when dealing with a static environment. Complications can arise when applying the basic PRM algorithm to dynamically constrained hosts in complex dynamic environments. Some of these complications are listed below.

1. Once the environment becomes dynamic, certain complications arise. One method of adapting the PRM is to compute each phase on-line, assuming that the algorithm is fast enough to model each instance in time as a static environment. This means that the algorithm runs as if, during each instant, the environment is static. This makes it possible to determine a path dynamically and create the roadmap if the environment changes.

2. Another factor that hinders the basic algorithm is that if sampling in time becomes impractical or inefficient and is removed, determining conflicts, on-line or during edge computation (Algorithm 1, line 10 and Algorithm 2, lines 5 and 8), becomes computationally intractable for dynamic environments. Each edge that connects to a vertex can result in a different arrival time (measured from q_{init}), this means that all possible arrival times have to be stored for each vertex.
   A graphical representation of the problem can be seen in Figure 4.4. Each possible path (in red) results in different arrival times (t) for vertex one (v_1). Therefore, in order to determine whether a conflict will occur at vertex v_1, each time instance, t_1 to t_6, will have to be checked for conflict. As the number of

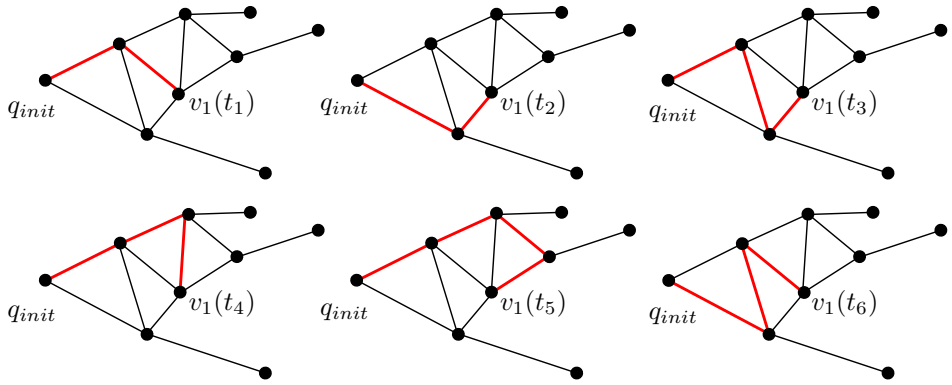


Figure 4.4: Multiple Arrival Times, due to not sampling in time.

vertices increases so does the number of potential arrival times at each vertex, thereby increasing the complexity of the algorithm.

3. Finally the graph structure of the PRM is impractical when attempting to determine a path for a host with differential constraints. Each edge that connects to a vertex and stems from it will affect whether the path is dynamically viable. This means that not all the paths that stem from a vertex will be dynamically reachable from the previous vertices. Figure 4.5 shows how, for a host that cannot

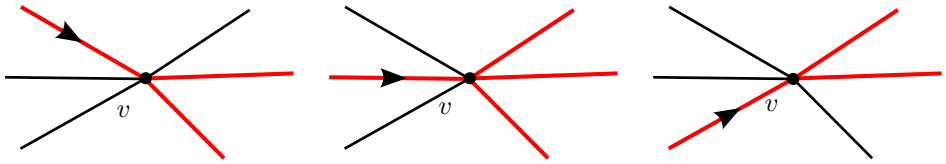


Figure 4.5: The different dynamically realisable edges (red) from vertex v based on the selected entry edge.

make sharp turns, only certain paths (in red) are viable at vertex v when entering from different edges. Implementing a graph structure on a host with differential constraints will require the algorithm to store all viable edge pairs at each vertex, making it computationally expensive to store and compute viable paths for dense roadmaps (roadmaps that contain numerous edge pairs for each vertex). This attribute makes it an impractical solution for non-holonomic systems (see Section 3.3) [64].
Furthermore, different routes taken to the same vertex will mean the uncertainty about the vehicle location at that vertex may be different, which means that the probability of conflict for a section of the path may be different based on the route taken to that section of the path. This is true for both static and dynamic environments, therefore a graph structure is not appropriate for motion planning in dynamic or uncertain environment.

4.6 Rapidly Exploring Random Tree

The RRT algorithm, first described by LaValle [64], is a single query path planner that can be used to traverse a space between a predefined initial and goal state (q_{init}, q_{goal}) [9]. The planner was initially created to function as a kinodynamic planner. This form of path planner attempts to determine a path that not only

traverses the C-space, but also adheres to the global obstacle constraints and local differential constraints of the host [71, 64]. The planners discussed to this point (Force field and PRM) are generally only viable for holonomic hosts (see Section 3.3). The RRT algorithm proposed here is capable of determining viable paths for non-holonomic hosts (see Section 3.3).

The main attribute of the RRT that makes this possible is the use of a tree structure when determining a path. The tree structure is advantageous because each vertex only stems from one edge. This makes it simple to apply differential constraints to a path. See Section 4.5.2 for the complications that can arise from general graphs in this regard. Another aspect that is favourable is that each node contains only one path to it from the root, therefore sampling in time is not required when attempting to determine conflict in a dynamic environment. The RRT algorithm described below proposes a non-holonomic path planner that explores the C-space quickly and can effectively be applied to systems with high degrees of freedom and complicated dynamics [71]. Finally it has been proven to be probabilistically complete [64]. These attributes make it a viable option as a conflict resolution algorithm. The basic functionality of the algorithm is described in the sections below.

4.6.1 Basic Algorithm

The basic RRT algorithm attempts to capture the connectivity of the C-space. It generally stems from a root placed at q_{init}. More complex algorithms build multiple trees, stemming from different locations in the C-space, as seen in Section 4.6.2. Commonly one root of the tree is placed at q_{init} and the other at q_{goal}. This means that the tree can be extended or grown towards the centre and joined to create a safe path from initial to goal state. The basic RRT algorithm can be described using three sub-algorithms: Build, Extend and Connect. The Build algorithm creates the tree, while the Connect algorithm is used to attempt a connection to q_{goal}.

Build Phase

The build phase of the RRT algorithm, described in Algorithm 3, creates a randomly generated tree. This is accomplished by randomly sampling the C-space and making use of the Extend Algorithm to add these vertices to the tree.

Algorithm 3 RRT - Build Phase, adapted from LaValle and Choset et al.[71, 9]

Input
$q_{init} \rightarrow$ the intial state and root of the tree

1: $V \leftarrow q_{init}$
2: $E \leftarrow \emptyset$
3: $T = (V, E)$
4:
5: **while** $|V| \leq n_S$ **do**
6: $n \leftarrow$ randomly sampled state in $\mathcal{C}$
7: $[v', T] \leftarrow$ extendRRT(T, n)

The algorithm starts by initialising an empty tree structure $T(V, E)$ and assigning the root of the tree to q_{init} [9, 72]. Then random states are sampled within the C-space, using the desired sampling strategy (see Section 4.3). Once a new state n is sampled, the algorithm adds it to the tree. This is accomplished by the Extend algorithm, depicted in Algorithm 4 and graphically in Figure 4.6.

Extend Phase

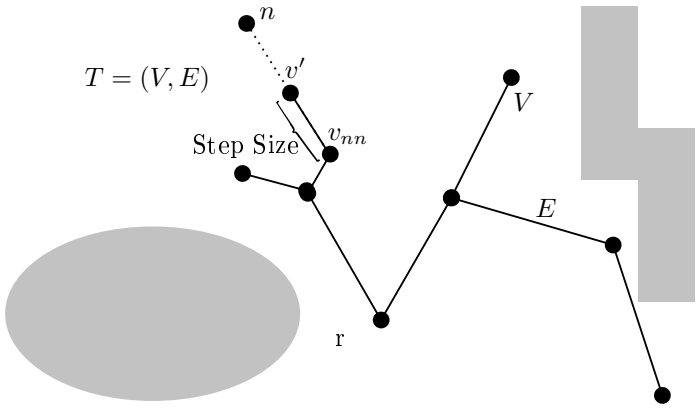


Figure 4.6: Extend RRT algorithm adds vertex v' to the tree, adapted from Choset et al. [9].

The extend algorithm adds vertices to the tree by attempting to move a predefined step size from the nearest neighbour towards the sampled state. If this is dynamically realisable and conflict-free, this new vertex v' is added to the tree. In order to ensure that the new samples are dynamically realisable a steer function (LPM) can be used, in Line 2 Algorithm 4, ensuring that the path from v_{nn} to v' is reachable (Section 5.6 details the functioning of an LPM that can be implemented, although numerous others exist). This process continues until the tree is populated by the desired number of samples n_S.

Algorithm 4 RRT - Extend Phase, adapted from LaValle and Choset et al.[71, 9]

Input
$T(V, E) \rightarrow$ a tree structure populated with vertices and edges
$v' \rightarrow$ the vertex to be added to the tree

1: $v_{nn} \leftarrow$ nearest neighbour to n, where $v_{nn} \in V$
2: $v' \leftarrow$ the state a step-size from v_{nn}, towards the random sample n
3:
4: **if** $(v_{nn}, v') \notin E$ and $\tau(v_{nn}, v') \neq \emptyset$ and $\tau(v_{nn}, v')$ is conflict-free **then**
5: $V \leftarrow V \cup v'$
6: $E \leftarrow E \cup (v_{nn}, v')$
7: $T = (V, E)$
8: **return** $[v', T]$
9:
10: **return** $[\emptyset, T]$

Due to the tree structure used by the RRT conflict detection can be executed while populating the tree. The conflict detection can be implemented even in dynamic environments, regardless of whether sampling in time is used. This is because each vertex in the tree has a individual unique path from the initial state or

root, resulting in a single time to each vertex which can be calculated immediately once the vertex has been added to the tree, assuming that the speed of the vehicle remains constant along the path. This individual path characteristic of the tree structure further simplifies the application of the algorithm to dynamically constrained hosts. These attributes are an attractive feature of the RRT, especially with regard to dynamic, differentially constrained path planning environments.

Connect Phase

Once a successful tree has been generated by the Build algorithm, a method is required to connect this tree to the goal state (q_{goal}) or goal region. The Connect algorithm uses the tree and attempts to link it to the goal. This is accomplished by repeatedly stepping the tree towards the goal state, through the use of the extend algorithm. The extend algorithm receives the complete tree T and the q_{goal} and it returns a vertex in the direction of the goal. This process is continued until the extend algorithm either reaches q_{goal} or it returns a failure. Failure can either be attributed to differential constraints or a predicted conflict. If a successful path is found then a search algorithm is used to extract the entire path (q_{init} to q_{goal}) from the tree.

Algorithm 5 RRT - Connect Phase, adapted from LaValle and Choset et al.[71, 9]

Input
$q_{goal} \rightarrow$ the goal or final state desired

repeat
    $[v', T] \leftarrow \text{extendRRT}(T, q_{goal})$
until $v' \in q_{goal}$ || $v' == \emptyset$

if $v' \in q_{goal}$ **then**
    return Success
else
    return Failure

4.6.2 Multiple Trees

The single tree generation method described in Section 4.6.1 can be ineffective when a single goal state is desired. Alternatively multiple trees can be generated and rooted at significant vertices to reduce the computation time. One such example is a bidirectional planning algorithm [71, 10]. This method grows two RRTs, one rooted at q_{init} and the other at q_{goal}. Both trees are grown independently and towards each other. Vertices are continuously added to both trees until a common vertex has been found. A common vertex is defined when $(v_i, v_g) \leq \epsilon$, where v_i and v_g are vertices from the trees stemming from q_{init} and q_{goal} respectively, given that ϵ is some small positive value [71]. A merge algorithm is used once a common vertex has been identified (see LaValle for more details on merge algorithms [10]). The algorithm merges the two trees at the common vertex and the final path can be extracted from the merged tree. Certain complications can arise when applying the bidirectional algorithm to dynamic, differentially constrained environments. Two of these problems are discussed below.

1. In a dynamic environment the generation of the initial tree (rooted at q_{init}) functions exactly the same as in the basic RRT example, but some fundamental complications can arise when creating the goal tree (rooted at q_{goal}). When populating the goal tree no conflict detection can be implemented, unless sampling in time is used. This is because there is no way to determine the time at which the host will reach any sample in the goal tree. This makes it an applicable method for dynamic environments only if sampling in time is implemented.

2. One favourable characteristic of the basic RRT algorithm is that it can be implemented for non-holonomic hosts, while multiple RRTs generally exhibit a discontinuity at the point of merger [71]. If this discontinuity can not be resolved by the host's LPM, it could pose a problem. An alternative technique is to translate the goal tree, placing v_g on v_i and thereby removing the discontinuity. However, this causes additional computation as the goal tree has to be rechecked for conflict, since all the vertices and edges have been shifted. This problem shows that bidirectional trees are not appropriate when dealing with uncertain environments.

4.6.3 Discussion of the Basic RRT

The basic RRT does not require the adaptation that the basic PRM required when applying it to a dynamic environment. This is because the RRT already requires the regeneration of the entire tree when a new root is selected. General RRT algorithms used for path planning or resolution problems do not make use of a final goal state, but instead a goal region. Pharpatara et al. made use of the technique in their paper "Sampling-based path planning: a new tool for missile guidance" [73], using a desired goal region and trajectory instead of a single state. This attribute has advantages and disadvantages with regard to conflict resolution. A goal region can reduce the computation time dramatically, since the completion criteria are not as strict as a single state. Alternatively, when applying resolution within an airport environment, declaring a large goal region could be problematic, due to the restricted airspace. Finally it can become difficult to ensure that the entire goal region is conflict-free. When developing the basic RRT algorithm there arise certain metrics and functions that can have large effects on the execution and computational efficiency of the algorithm. Some of these are listed below.

1. An unfortunate feature of the general RRT algorithm is that it will never strive towards an optimal solution. Karman and Frazzoli state that as the number of samples tends towards infinity, the paths determined by the RRT algorithm, under reasonable technical assumptions, do not tend towards an optimal solution [62].

2. When the RRT algorithm determines a nearest neighbour it selects the desired vertex according to a predefined metric. LaValle and Kuffner found that the performance of this selected metric had a large effect on the algorithmic computation time [71]. They also concluded that the process of determining the ideal metric was just as complicated as solving the entire path planning problem. Having the algorithm's performance so dependent on a single metric selection can pose a problem, since a conflict resolution system has to be robust for multiple unique scenarios.

3. Computation time is of paramount importance and in order to determine whether the RRT algorithm can be implemented in a practical situation, all potential bottlenecks have to be identified. The nearest

neighbour search is one such bottleneck [71]. Other computationally expensive or inefficient characteristics identified are: data structures used, nearest neighbour metric and the effect of different C-spaces. In order to ensure effective and efficient path generation, the RRT algorithm must be adapted for the proposed application.

4. Finally the selection of the step size (maximum edge length) affects the exploration of the C-space as well as the number of samples. The selection of this parameter, in high-dimensional scenarios, results in large trade-offs between the two [9]. By selecting a large step size, fewer samples are required, but the exploration resolution is low resulting in the possible omission of important regions in the C-space. Dynamically adapted step sizes can be used to apply the same algorithm to multiple scenarios. One method is to base the step size on a function of the euclidean distance between q_{init} and q_{goal} [9]. Another method is to remove the step size entirely and connect the sampled vertex directly to the tree [10].

The RRT algorithm is an efficient and fast sampling-based technique that can be applied to a resolution problem [71]. Its effective implementation on a non-holonomic, dynamically constrained host is crucial, with regard to aircraft applications. The basic RRT algorithm has an exponential decay of the probability of failure. This implies that the algorithm converges on a solution (if one exists) at an exponential rate, as the number of samples increases [62]. Finally it allows for conflict detection while creating the graph, without sampling in time, saving computation time and reducing complexity.

4.7 Incremental Sampling Algorithms

Incremental sampling is an adaptation of the general sampling methods described in Section 4.2. The difference occurs at steps 2 and 3. Basic sampling methods first populate a graph and then search through it for a viable path, while incremental sampling creates and searches through the graph at the same time (see Algorithm 6). This is accomplished by continuously sampling a state (n) and adding it to the tree until a path is found to the goal state or some termination criterion is adhered to. Therefore, while the tree is growing, checks are being done to determine if the goal state or region has been reached.

Incremental sampling is a single query search, meaning that for each new initial and goal state (q_{init}, q_{goal}) a new search tree has to be populated [10]. Multiple query searchers require an off-line predetermined search tree (see Section 4.5.2). The PRM algorithm in Section 4.5 is an example of a multi-query search and cannot be implemented as an incremental sampler. The RRT algorithm presented in Section 4.6 can, however, be applied as a incremental path planner and consequently is considered to be a method of incremental sampling [10]. The algorithm creates an undirected search graph to capture the connectivity of the C-space. Implementation as a non-holonomic path planner requires the application of an action space that ensures that the differential constraints of the host are adhered to (see Section 5.6.1) [10].

There are many different methods of incremental sampling; some of these are listed below. Each is an adaptation of the basic incremental sampling method depicted in Algorithm 6.

Algorithm 6 Incremental Sampling, adapted from LaValle [71]

Input
$q_{init} \rightarrow$ the root of the tree
$q_{goal} \rightarrow$ the goal or final state desired

$V \leftarrow q_{init}$
$E \leftarrow \emptyset$
repeat
  $n \leftarrow$ randomly sampled state in $\mathcal{C}$
  $v_{cur} \leftarrow$ select a vertex $|v_{cur} \in V$
  if $n \notin V$ and $(v_{cur}, n) \notin E$ and $\tau(n, v_{cur}) \neq \emptyset$ and $\tau(n, v_{cur})$ is conflict-free **then**
    $V \leftarrow n \cup V$
    $E \leftarrow (v_{cur}, n) \cup E$
until $(q_{init}, q_{goal}) \subset E$ ▷ q_{goal} can be a region in $\mathcal{C}$

4.7.1 Randomised Potential Field

The randomised potential field method [63] is a hybrid between the basic Force Field methods proposed in Section 2.3.1 and random sampling. It makes use of the potential field functionality of the Force Field methods by defining attractive and repulsive terms based on some desired metric (generally Euclidean distance). The definition of these terms requires many finely tuned, scenario dependent heuristics that impact the effectiveness of the planner. This is used to solve the path planning problem. In order to avoid the local minima problem that is prevalent in many Force Field algorithms, an additional random walk functionality is added to the planner. When the algorithm determines that it could be tending towards a local minima it makes use of a random walk to escape. This random walk is determined by using general sampling-based techniques to generate random escape paths. The algorithm is described in depth by LaValle [10] and Latombe [63]. This method has been shown to be very effective with excellent results with up to 31 degrees of freedom. Unfortunately the planner is highly dependent on the selection of accurate heuristics. These specific scenario heuristics make it impractical to apply the method as a robust scenario-independent resolution system.

4.7.2 Ariadne's Clew Algorithm

This algorithm attempts to grow the search tree by biasing the new samples to explore as much new space as possible [10]. It aims to determine a viable path while also collecting as much information as possible about the C-space [74]. This method focuses on investigating the entire C-space by investing time in global exploration. The main disadvantage of this algorithm is that it is very difficult to optimise the sampling method. Genetic algorithms have been applied, but generally these methods are avoided for path planning problems due to their dependence on scenario specific parameter tuning [10].

4.7.3 Expansive-Space Planner

This planner is similar to the Ariadne's Clew algorithm described above. It also biases its samples to promote the exploration of the C-space and attempts to capture the global connectivity of the environment.

Additionally, however, the algorithm makes use of a bidirectional search method to improve the planner's efficiency [10]. Vertices are selected (v_{cur}) (see line 5 of Algorithm 6) with a probability that is inversely proportional to the number of vertices surrounding it within a predefined radius. Therefore the algorithm biases its growth towards areas that are less populated. This planner solves multiple problems by using this relatively simple sampling criterion [10, 75]. The definition of the search radius determines the resolution of the search algorithm and its performance tends to degrade when searching through a long labyrinth. Finally the algorithm requires extensive, problem specific parameter tuning, making it impractical to apply to scenario-independent path planning [10].

4.7.4 Random Walk Planner

The random walk planner essentially functions as the second section of the randomised potential field planner. It makes use of entirely random walks to determine a search graph that captures the connectivity of the space. Parameter tuning is avoided by adjusting the sampling directions and radius after each iteration, based on the number of previously sampled states [10]. Each newly sampled state is connected to the previously sampled state, if dynamically possible. The primary downfall for the algorithm is its inability to effectively determine a path down a long passage. Additionally the paths generated require post-processing in order to smooth out unevenness that can occur along it [76]. Generally the algorithm is used in conjunction with other methods of path planning as seen with the random potential field method [10].

4.7.5 RRT*

The RRT* developed by Karaman and Frazzoli is an incremental sampling-based adaptation of the RRT algorithm proposed in Section 4.6 [62]. The proposed algorithm inherits the probabilistic completeness and exponential decay of probability of failure from the RRT. Additionally the adaptations made ensure that the paths determined now are asymptotically optimal. The RRT and RRT* have the same Build phase (see Algorithm 3) they only differ with respect to the Extend phases. The RRT* adapts its extend phase to ensure asymptotic optimality. This is accomplished by attempting to connect a new vertex to all states within a predefined radius, selecting the optimal connection, instead of a single connection (as with the RRT). Additionally the algorithm remaps the edges of all the nearest vertices to the newly added vertex based on their costs.

The extend phase (see Algorithm 7) starts by determining the nearest vertex (v_{nn}) to the newly sampled state (n) and stepping towards it based on the predefined step size. The algorithm then tests all the vertices within a certain range and determines the one that contains the minimum cost, from the initial state q_{init}. This cost is determined using the function $C(v)$, which determines the cost of the unique path from q_{init} to v. The new vertex is then linked via an edge to the lowest costing vertex (see Figure 4.7a). If this is not possible the next lowest costing vertex is linked, until a connection is made or all the vertices in the range have been tested.

The basic RRT method undergoes further adaptations during the second section of the Extend algorithm where the graph is reshaped to ensure that all connections in the tree are optimal. The reshaping of the tree

Algorithm 7 RRT* - Extend Phase, adapted from Karman and Frazzoli [62]

Input
$T(V, E) \rightarrow$ a tree structure populated with vertices and edges
$n \rightarrow$ the vertex to be added to the tree

$v_{nn} \leftarrow$ nearest neighbour to n, where $v_{nn} \in V$
$v' \leftarrow$ the state a step size from v_{nn}, towards the random sample n
if $\tau(v_{nn}, v')$ is conflict-free **then**
    $V \leftarrow V \cup v'$
    $v_{min} \leftarrow v_{nn}$
    $V_{near} \leftarrow$ nearest vertices within a predefined range of $V' \mid V_{near} \subset V$
    for each $v_{near} \in V_{near}$ **do**
        if $\tau(v_{near}, v')$ is conflict-free **then**
            $c \leftarrow C(v_{near}) + c(\tau(v_{near}, v'))$
            if $c < C(v_{min}) + c(\tau(v_{min}, v'))$ **then**
                $v_{min} \leftarrow v_{near}$
    $E \leftarrow E \cup (v_{min}, v')$
    for each $v_{near} \in (V_{near} \setminus v_{min})$ **do** ▷ This implies that v_{near} is an element of V_{near} excluding v_{min}
        if $\tau(v', v_{near})$ is conflict-free and $C(v_{near}) > C(v') + c(\tau(v', v_{near}))$ **then**
            $v_{stem} \leftarrow$ the vertex in V that connects to v_{near}
            $E \leftarrow E \setminus (v_{stem}, v_{near})$
            $E \leftarrow E \cup (v', v_{near})$
    $T = (V, E)$

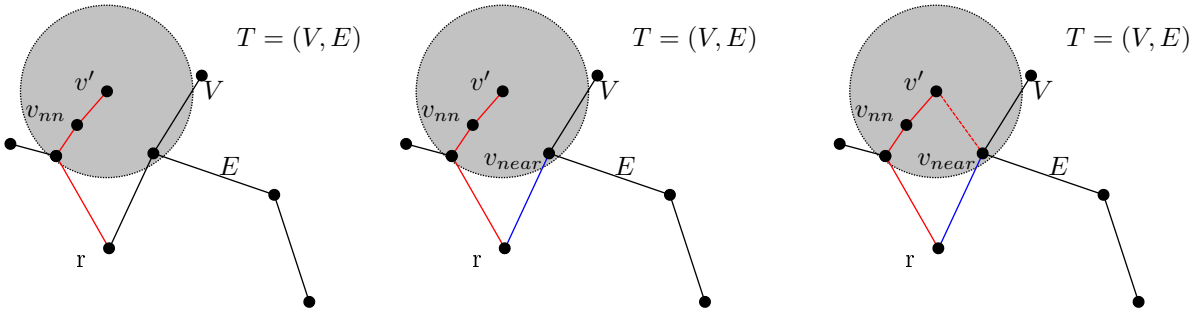
return $[\emptyset, T]$

is accomplished by first determining the current cost of all the vertices within range of the v' represented by: $C(v_{near}) \mid v_{near} \in V_{near}$, where V_{near} is a set of all the vertices within the range of v'. This is shown graphically as the blue path in Figure 4.7b. Then these costs are compared to the cost from q_{init} via v' to v_{near} (see the red path in Figure 4.7c) described by: $C(v') + C(\tau(v', v_{near}))$, where τ denotes a path and $C(\tau(e))$ is the cost of the path described by the edge e. If the new cost is less, then the edge to v_{near} is remapped via v'.

A graphical representation of this reshaping process is shown in Figure 4.7. The process starts once a final path to v', the red path in Figure 4.7a, has been determined. Then all the paths in range, the grey circles, are identified as part of V_{near}. The paths to each vertex in V_{near} are identified; the blue path in Figure 4.7b represents one such path. Finally the cost of the blue path is compared to the red path in Figure 4.7c and the tree is remapped according to the lowest costing path.

4.7.6 Discussion of Incremental Sampling

The development of incremental sampling techniques stemmed from the desire to create a probabilistically complete, single query, real-time path planner [62]. Of the algorithms described above the, RRT and RRT* can be considered as the most desirable with regard to this aim. Different incremental sampling approaches can have many favourable characteristics, some of which are described below.



(a) Path identified (red) from r to v' in $T = (V, E)$

(b) Additional path (blue) determined, where $v_{near} \in V_{near}$

(c) Path (red) determined from r to v_{near} via v'

Figure 4.7: Reshaping process of the RRT*, where all paths within range (the grey circle) are tested and the best costing paths are identified and remapped

1. The generation of dynamically viable paths. Incremental sampling methods can be implemented using a tree structure, which makes them applicable to non-holonomic hosts. Additionally the tree structure simplifies conflict detection, without sampling in time, due the generation of unique paths to each vertex.

2. On-line implementation of some methods is possible due to their computational efficiency, allowing for real or near real-time execution. This makes the systems capable of handling static and dynamic environments.

3. Some incremental sampling algorithms, like the RRT*, have been proven to be asymptotically optimal [62]. This is a very attractive attribute, especially with regard to path planning and conflict resolution. It implies that the resolution paths found will strive towards an optimal solution.

An incremental sampling approach is a fast and effective adaptation of the general sampling methods that possess many favourable attributes with regard to conflict resolution and path planning. This makes them an attractive selection for a potential conflict resolution system.

4.8 Summary

Sampling-based path planning techniques show numerous desirable characteristics when applied as conflict resolution systems. The probabilistic completeness of the algorithm coupled with the fast execution times promote the fast generation of safe resolution paths. The different sampling-based methods each contain numerous advantageous characteristics; of these, the RRT* algorithm is the most favourable. This algorithm has been proven to be asymptotically optimal, ensuring that the algorithm strives to derive an optimal solution. Additionally RRT methods have successfully been applied as kinodynamic path planners. This implies that the algorithm can cater for dynamically constrained vehicles. The algorithm's ability to derive fast resolution paths, while also striving towards an optimal path, ensures that it adheres to both the primary and secondary requirements stated in the problem statement. This makes the RRT* algorithm a promising method for application as a conflict resolution system.

Chapter 5

System Implementation

5.1 Overview

The algorithm described in this chapter has been developed to function as a conflict resolution system for an aircraft in an airport environment. The primary aim of the system is to, as quickly as possible, determine a conflict-free path that avoids the conflict region and reaches a desired goal state. Once this has been achieved, optimisation of the path, based on a predefined cost function, can be attempted. In Chapter 3 we described in depth the challenges that arise due to the application of the algorithm to an aircraft in an airport environment. When implementing it as a decentralised system on an aircraft (as described in Section 2.1.3 for details) certain requirements have to be satisfied. The system has to perform the desired path calculations at real-time or near real-time to ensure that it can effectively handle short term collisions. Therefore a reliable, computationally effective algorithm is required. Furthermore, the dynamic constraints of the aircraft restrict the potential path calculations. The determined resolution paths have to be differentially realisable to ensure that they are indeed flyable. This requires a dynamically constrained resolution algorithm (see Section 3.1). The airport scenario adds computational complexity, because it requires that the system cater for both dynamic and static obstacles in a relatively cluttered environment. Descriptions of these requirements and potential complications are discussed in depth in Chapter 3.

We have identified sampling-based techniques as the best fit family of resolution methods based on these requirements. Sampling-based techniques have been proven to be computationally efficient as well as probabilistically complete (see Section 2.3.2.1). These attributes make it very attractive. In Chapter 4 we reviewed some potential sampling-based algorithms, namely the PRM, RRT and Incremental Sampling. The former two algorithms are very effective but do not ensure an optimal solution as the number of samples tends towards infinity (is described in Section 4.7). Finally the Incremental Sampling techniques have many favourable characteristics, especially the RRT*, described in Section 4.7. This algorithm has been proven to be probabilistically complete as well as asymptotically optimal.

In the sections below we discuss our implementation of an adapted RRT*. We start by highlighting the assumptions and design choices made due to the practicalities that arise when applying the algorithm to

a real world aircraft environment, in Section 5.2. Then we move on to the description and algorithmic implementation of the adapted RRT*, in Section 5.3. This section describes the adaptations made to the RRT and RRT* algorithms described in Sections 4.6 and 4.7. The adapted RRT* algorithm makes use of numerous internal subsystems to ensure that the final path determined is dynamically feasible as well as optimal. The LPM module discussed in Section 5.6 generates paths that adhere to the differential constraints of the aircraft, thereby ensuring that all paths determined by the resolution system are flyable. The optimisation of a path is performed based on some predetermined cost function. A description of what a cost function entails as well as the calculation of the different cost functions applied to the system can be found in Section 5.7. We conclude the chapter with descriptions of the different search optimisation techniques used to improve the computational efficiency of the algorithm in Section 5.8.

5.2 Practical Application to an Aircraft

The implementation of a path planner as a conflict resolution system for an aircraft requires real-time or near real-time computation. The proposed resolution system issues a resolution advisory to the pilot. However if no action is taken to follow the resolution path the system is capable of controlling the aircraft, in order to execute the required manoeuvres. In an attempt to achieve this computational efficiency we have made some assumptions and design choices, based on the practical aspects of an aircraft and the expected application of the system. These choices simplify the resolution problem and promote faster execution times.

5.2.1 Assumptions and Design Choices

It is assumed that a conflict detection module is available for use by the conflict resolution system. The application of the system to uncertain environments requires the application of a probabilistic conflict detection module that will handle all uncertainty (see Section 3.2.3 for justification of this assumption). Additionally we assume that the detection module will execute quickly and effectively, while still providing reliable outputs regarding the probability of conflict. All outputs provided by the detection module will be trusted and regarded as the truth. The simulation module used by the conflict detector (see Section 5.4) is expected to accurately predict the mean trajectory of the desired path based on an accurate model of the host aircraft.

We also assume that the aircraft will follow the proposed resolution path accurately, or at least with an error equivalent to the uncertainty assumed by the conflict detection module. Therefore all RAs supplied are expected to be accurately adhered to by the pilot. We assume that the aircraft is equipped with the required control system (see Section 3.1.4) to accurately track any dynamically realisable path proposed by the system. This path proposed to the pilot is the path determined by the resolution system after it has been simulated forward in time, ensuring that the path flown coincides with the path tested for conflict.

An important assumption made with regard to the functionality of the resolution module is that the aircraft maintains a constant speed. This assumption vastly simplifies the development of dynamically realisable paths, as well as the ability to predict the time at each vertex or waypoint. The assumption is made on the basis that we have developed a short term resolution system (see Section 2.1.3), and therefore large changes in speed are not expected.

We made a design decision to decouple the vertical and horizontal resolution systems. This implies that a resolution path cannot contain both vertical and horizontal manoeuvres. Therefore the vertical and horizontal resolution systems are run separately, but in parallel. The partitioning of the resolution system into separate vertical and horizontal path searches imposes a constraint that the initial state q_{init} and the goal state q_{goal} must be at the same altitude. If not, an alternative coupled LPM module will have to be developed to cater for climbing or descending flight plan. The decoupling of the domains simplifies the implementation of the LPM, promoting faster computation. Additionally the decoupled system results in resolution advisories that are easier for the human pilot to execute. Unfortunately the decoupled system also vastly reduces the search space and therefore scenarios may arise where the system could fail to find a path.

Finally we define the selection of the goal and initial states. The initial state of the system will be selected as the current location at the time that a conflict is detected. When implemented practically the initial state should be placed at least 5 seconds into the future along the current flight plan, to ensure that the pilot has sufficient time to react to an RA. This is based on the reaction time allowed by TCAS in Section 2.2.2. Two selection methods for the goal state have been identified. When a flight plan is available the goal state will be placed in the future along the flight plan, while in a controlled airport environment it is assumed that the ATC will designate a safe goal state. The goal states must be placed within the same horizontal plane as the initial states. This is to ensure that the dynamics of the aircraft are adhered to, as the decoupling of the LPM requires this assumption to apply horizontal resolution.

5.2.2 Axis Redefinition

The application of the system to real world scenarios requires the definition of a relevant frame of reference for algorithmic functionality. We have defined an XYZ-axis system that can be calculated from the inertial axis system described in Section 3.1. We made use of

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \Psi_{\mathbf{ax}} \begin{bmatrix} X_I \\ Y_I \\ Z_I \end{bmatrix}, \tag{5.1}$$

where $\Psi_{\mathbf{ax}}$ is a rotation matrix to rotate XYZ-axis onto the current trajectory in inertial coordinates. The rotation matrix $\Psi_{\mathbf{ax}}$ is defined as:

$$\Psi_{\mathbf{ax}} = \begin{bmatrix} \cos\psi_{gt} & \sin\psi_{gt} & 0 \\ -\sin\psi_{gt} & \cos\psi_{gt} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.2}$$

where ψ_{gt} is the ground track heading angle between q_{init} and q_{goal} at the time the resolution module is activated. The rotation equates to rotating the XYZ-axis so that the X-axis corresponds to the current ground track trajectory of the aircraft, as shown in Figure 5.1. The Y-axis direction then represents the cross track error direction, and the Z-axis remains the down axis in inertial coordinates.

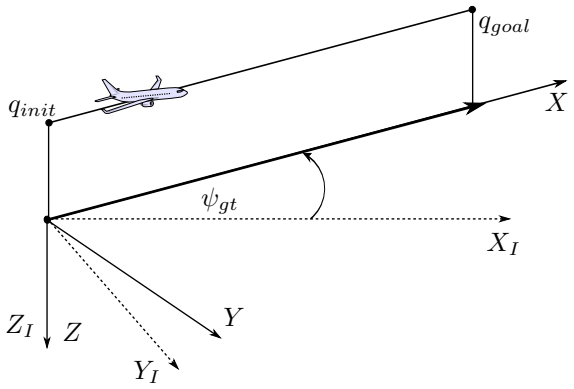


Figure 5.1: Rotation of the horizontal axis from inertial to XYZ coordinates

5.3 Adapted RRT*

The algorithm proposed in this section is an adaptation of the RRT and RRT* algorithms (described in Sections 4.6 and 4.7). The original algorithms have many favourable attributes, especially with regard to an aerospace application. We have adapted them in order to further improve on their effectiveness and efficiency as an aircraft conflict resolution system. The changes made to the original RRT and RRT* are listed below.

1. A new sorting method that connects the newly sampled states to the tree.
2. The step size originally used in the RRT [64] algorithm has been removed, favouring a direct sample connection method [10],
3. Finally, the RRT* algorithm can improve the paths in the tree, by reordering the vertices after each new sample is connected. This functionality has been removed to allow the application of a probabilistic conflict detection module. If any changes occur along a previously tested path, the probability of conflict along those paths has to be recalculated. This implies that after each reordering process the paths changed will have to be rechecked for conflict, which is a redundant and computationally expensive process [7].

Two important characteristics of the general RRT and RRT* algorithms that have to be inherited by the adapted RRT* algorithm in order to ensure that an optimal path can be found are probabilistic completeness and asymptotic optimality. The asymptotic optimality of an algorithm can be assumed if it adheres to the following requirements [62].

1. An additive cost function is used. This implies that for all paths the sum of the cost of each individual path is equivalent to the cost of their combined path.
2. The cost function implemented is continuous, therefore all paths in close proximity have similar costs.
3. The obstacle spacing is such that there exists some space around the optimal path that allows for convergence.

These assumptions are adhered to by the adapted RRT* algorithm, therefore the algorithm inherits the RRT*'s asymptotic optimality [62]. This implies that the adapted RRT* will strive towards an optimal solution as the number of samples tend towards infinity. However, asymptotic optimality does not mean that a path will be found as the number of samples tend towards infinity. In order to ensure this the algorithm has to be probabilistically complete. The changes made to the RRT to produce the RRT* are similar to the changes made to the RRT* to produce the adapted RRT*, therefore we assume that the adapted RRT* will inherit the probabilistic completeness from the RRT*. The tests applied to the adapted RRT* in Section 6.2.2 empirically support this assumption, however additional research is required in order to prove it. Search optimisation techniques are applied in an attempt to improve the computational efficiency of the path calculations. The adapted algorithm does not make use of the RRT*'s remapping function, but does use a cost-based sorting function to ensure that, if the sampled state can be added to the tree, then the most optimal path connection is made. The tree structure T is rooted at q_{init} and functions as a general RRT or RRT* tree.

5.3.1 Algorithm

Adaptations were made to the Build, Connect and Extend Phases of the RRT and RRT* algorithms. The Build and Connect phases explained in Section 4.6, Algorithms 3 and 5 have been merged to form a single Build and Connect Phase seen in Algorithm 8.

During this phase the states are incrementally sampled and connected to the tree T using the Extend phase in Algorithm 9. If a sampled state is successfully added to T then the algorithm attempts to connect the sample to the goal state q_{goal}. A successful goal state connection invokes the search optimisation function that selects the lowest costing path to goal. This process continues until a predefined termination criterion is achieved. Generally this is when the desired number of samples n_S has been reached. Finally the optimal path to goal P is extracted from the tree and returned.

The Extend Phase of the RRT* (Algorithm 7) has been adapted by removing the remapping functionality and adding a cost sorting function, as shown in Algorithm 9. The aim of the Extend phase is to determine an optimal connection between the newly sampled state v' and a state in V. This is accomplished by first identifying a set of all the dynamically flyable edges E' between the states in V and v' using the LPM motion planner described in Section 5.6. E' is then sorted in ascending order based on the cost from the root of the tree to the newly sampled state v' via each edge in E'. Finally the algorithm iterates through each edge until a conflict-free path is found. The path found describes the optimal conflict-free, flyable path from the root of the tree to the newly sampled state.

Once this path has been determined the algorithm returns the connecting edge e_{min}. If no conflict-free edge is determined the algorithm returns an empty set. This implies that no flyable conflict-free path was found from any state in V to the newly sampled state.

5.3.1.1 Parallel Execution

The decoupling of the vertical and horizontal domain requires the development of two search trees. If a solution only exists in one domain, it becomes inefficient to create the trees sequentially. Therefore the

Algorithm 8 Adapted RRT* - Build and Connect Phase

Input
$q_{init} \rightarrow$ the root of the tree
$q_{goal} \rightarrow$ the goal or final state desired

1: $V \leftarrow q_{init}$
2: $E \leftarrow \emptyset$
3: $T \leftarrow (V_{init}, E_{init})$
4: **for** $i = 1$ to n_S **do**
5: $n \leftarrow$ sampleState()
6: $e' \leftarrow$ extendRRT*(V, n)
7: **if** $e' \notin E$ and $e' \neq \emptyset$ **then**
8: $E \leftarrow E \cup e'$
9: $V \leftarrow V \cup n$
10: $T \leftarrow (V, E)$
11: $e' \leftarrow$ extendRRT*(n, q_{goal})
12: **if** $e' \neq \emptyset$ **then**
13: $V \leftarrow V \cup q_{goal}$
14: $E \leftarrow E \cup e'$
15: $E' \leftarrow E \cup e'$
16: $T \leftarrow$ optimiseTree(V, E, e')
17: $P \leftarrow$ finalPath(T)
18: **return** P

adapted RRT* is modified so that it builds the search trees in parallel, as seen in Algorithm 10.

The horizontal and vertical domains are represented by the two tree structures $T(1)$ and $T(2)$. The adaptedRRT* function essentially executes lines 5-11 of Algorithm 8, returning the updated tree ($T(j)$) and the edge that connects to the goal state (e_{goal}) if one is found. Algorithm 10 is implemented to ensure that during each iteration a single vertex is added to both search trees, allowing the parallel growth of each tree. When a path to the goal state is determined in either domain (vertical or horizontal), the search optimisation functions are called (see Section 5.8). These search optimisation techniques (optimiseTree) determine the optimal search trees in both domains. The implementation of parallel execution allows the the application of cross dimensional optimisation, described in Section 5.8.4.

5.4 Interaction Between CD&R systems

The interaction between the different modules within the CD&R system is important to understand. This section describes how the two systems interact within the currently developed architecture. The conflict detection module is used within the extend phase (Algorithm 9) to determine whether the path described by an edge is conflict-free. This interaction is depicted in the form of a flow diagram in Figure 5.2. The conflict resolution module is activated (Begin) when the outer loop conflict detection system predicts a conflict along the current trajectory. The resolution module makes use of Algorithms 8 and 9 to determine a safe path.

A safe path is determined by first populating a search tree with edges and vertices. Before each edge is added to the tree it must be checked for conflict. This is achieved by sending the path segment described

Algorithm 9 Adapted RRT* - Extend Phase

Input
$V \rightarrow$ a set of all the vertices to be connected
$v' \rightarrow$ the vertex to be added to the tree

$E' \leftarrow \text{LPM}(V, v')$ ▷ Returns a set of all the edges from V to v'
if $E' \neq \emptyset$ **then**
  $E_{sort} \leftarrow \text{sort}(E')$
  $e_{min} \leftarrow$ select shortest path in E_{sort}
  for $i = 1$ to $|E_{sort}|$ **do**
    if e_{min} is conflict-free **then**
      return e_{min}
    else
      $e_{min} \leftarrow$ next shortest path in E_{sort}
return $\emptyset$

Algorithm 10 Adapted RRT* - Parallel Execution

Input
$q_{init} \rightarrow$ the root of the tree
$q_{goal} \rightarrow$ the goal or final state desired

for $j = 1$ to 2 **do**
  $V(j) \leftarrow q_{init}$
  $E(j) \leftarrow \emptyset$
  $T(j) \leftarrow (V(j), E(j))$
for $i = 1$ to n_S **do**
  for $j = 1 \rightarrow 2$ **do**
    $(T(j), e_{goal}) \leftarrow \text{adaptedRRT}^*(T(j), q_{goal})$
    if $e_{goal} \neq \emptyset$ **then**
      $V(j) \leftarrow V(j) \cup q_{goal}$
      $E(j) \leftarrow E(j) \cup e_{goal}$
      $T \leftarrow \text{optimiseTree}(V, E, e_{goal})$
$P \leftarrow \text{finalPath}(T)$
return P

by an edge to the conflict detector. The path segment can either be encoded as a set of manoeuvres or an array of sampled vertices or waypoints indexed in time, depending on the requirements of the detection module. The detection module makes use of an environment and aircraft model to extract and predict the current and future states of the system. These simulation models utilise the aircraft dynamics in conjunction with the desired ideal path proposed by the resolution module to determine the simulated mean trajectory of the aircraft as well as its predicted standard deviations. This information is fed to the conflict detection module and used to determine whether a conflict will occur along the path segment. The detection module then returns either a binary or probabilistic chance of conflict. The proposed external probabilistic detection module will return the PDF or Cumulative Density Function (CDF). The resolution module will then make use of some threshold to determine whether the path segment is indeed conflict-free as seen in Figure 5.2. This process is continued until a predefined termination criterion is reached. Then the final path is proposed. If the final path is conflict-free, the flight plan is updated and an RA is issued. If not, then the resolution module is reactivated and the entire process is repeated until a safe path is found.

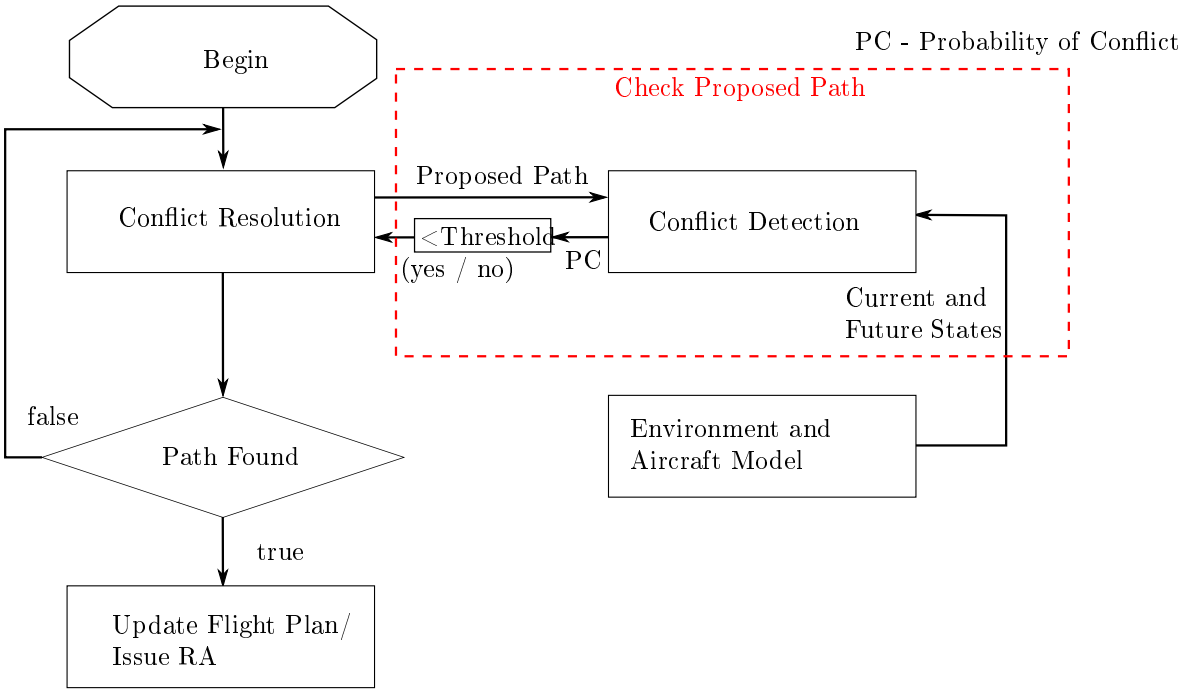


Figure 5.2: A flow diagram depicting the internal interaction between the conflict resolution and detection modules

5.5 Bounding the Configuration Space

The configuration space of an aircraft is an infinitely large region, restricted only by the differential capabilities of the aircraft. Additionally the C-space is made up of uncountably many states, making it computationally impractical to attempt to sample the entire C-space. Therefore we require a method of limiting the C-space. This means that the sampler only samples a bounded subset of the entire C-space.

For the purposes of path planning, an aircraft can be modelled using 6 states $(x, y, z, \theta, \psi, \phi)$, three coordinate and three attitude parameters. We add an additional time state to the system, in order to implement dynamic conflict detection (see Section 4.4). Therefore we have a 7 dimensional C-space. In order to reduce this we have made the following practical simplifications. The control of the attitude parameters is managed by the LPM function. Due to the non-holonomic characteristics of the host, the attitude parameters of the system can be determined from the path taken to each state (see Section 3.3). Therefore we do not require the sampling of the attitude characteristics. The system will determine the final attitude parameters of the state once the path to the state has been calculated. This reduces the computational complexity of the sampler, because a complex sampling strategy is not required to sample the attitudes effectively.

The states are further reduced by not sampling the time domain. This is done to reduce the computational complexity of both the LPM and the sampler. Sampling in time is used in conjunction with the conflict detection module to help solve for dynamic environments. Additionally it helps determine paths for systems where a safe path only exists over a certain time period. The use of a tree structure helps remove the need for sampling in time with regard to the conflict detection module, as each vertex has a unique path from the root. This means that the time to each vertex can be determined based on these unique paths. The nature of an

aircraft scenario implies that an 'open door problem' will never practically occur during short-term conflict resolution. The sampling of the time domain can result in complications for the LPM function. If sampling in time is implemented, then in order to successfully connect each state, the host will have to adapt its speed accordingly. Apart from speed changes being the most inefficient resolution method, it also implies that the LPM function will have to cater for changes in speed and manage the upper and lower thresholds. This is simple when considering a single straight and level manoeuvre, but can become complex when combined with climb, descent and banking manoeuvres. Finally, changes in speed to ensure that the arrival time at each vertex is adhered to, can be fuel inefficient.

The sampler proposed in this chapter will therefore only sample the position parameters of the aircraft and make use of the tree structure and a more complex LPM to determine the additional attitude and time parameters at each vertex.

5.5.1 Sampling the C-Space

The coordinate based C-space can be described as any location that the aircraft can possibly reach. This means that this state space is still represented by an uncountable number of states. Therefore practical implementation requires the reduction of this coordinate C-space. The limited C-space aims to incorporate all states that could possibly yield a favourable result. Additional functionality can be added to ensure that all states sampled are indeed dynamically reachable (see Section 5.8.2) [7].

Determining the sample space size can be crucial to the efficiency and effectiveness of the resolution system. The selection of a too large sampling region can reduce the efficiency of the algorithm and unnecessarily increase the computation time. Alternatively, restricting the sampling region too much can result in no path being found. We require an effective method to select the sampling region size that ensures that a path will be found, without unnecessary computation. An effective approach that makes use of a constant speed assumption or maximum speed constraint is the use of an ellipsoidal sampling region [14]. The size of the ellipsoid is determined as a function of some constant speed. This constant speed can be derived from the maximum aircraft speed or a constant speed model can be assumed. The sample region can be based on either, but for the purpose of this application we assume a constant speed model.

5.5.1.1 Elliptical Sampling Region

The selection of an elliptical sampling region is based on its desirable distance characteristics. The straight-line distance between the foci of an ellipse via any point on the ellipse's boundary remains constant, therefore the straight-line distance between foci and any point within the ellipse is less than this constant value. This ensures that no samples are generated that will result in a path length larger than the maximum constant value defined by the boundary of the ellipse. The derivation of the ellipsoidal sampling region requires the placement of the initial and final states on the foci of the ellipsoid. Additionally the vertical and horizontal generation of the sampling region is decoupled (see Section 5.2) to simplify the LPM calculations as well as the sampling methods. The resultant sample spaces can be represented as two elliptical regions. The boundaries of these regions are defined based on the dynamic capabilities of the aircraft. The maximum

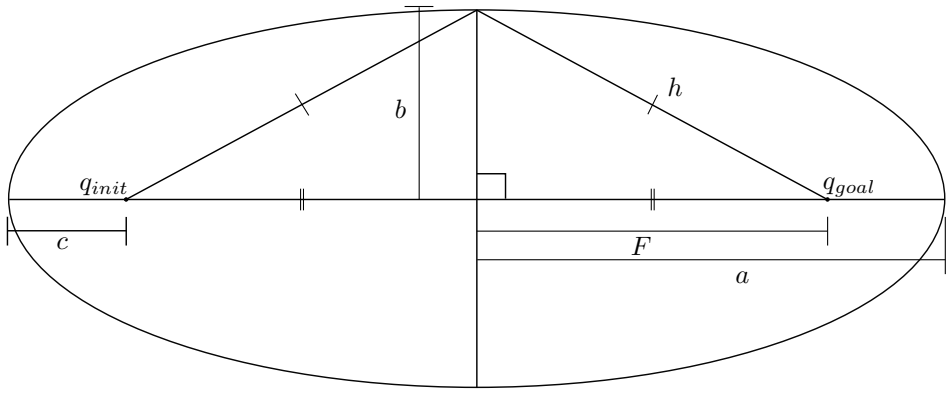


Figure 5.3: Construction of the elliptical sampling region using a constant speed constraint

constant distance is the distance between the two foci via any point on the boundary of the ellipse and can be determined, from Figure 5.3, as:

$$2c + 2(a - c) = 2a, \tag{5.3}$$

where a denotes the semi-major axis of the ellipse and c is the distance from the foci to the end of the ellipse. Equation 5.3 shows that the distance h in Figure 5.3 is equal to the semi-major axis a. This constraint on the maximum distance allows the selection of either the semi-minor or semi-major axis and the calculation of the other based on Equations 5.4 and 5.5.

$$a = \sqrt{b^2 + F^2} \tag{5.4}$$

$$= \sqrt{b^2 + |q_{goal} - q_{init}|^2} \tag{5.5}$$

Equation 5.5 allows us to calculate the spatial dimensions of the ellipse by selecting either axis a or b. The selection of the semi-minor axis for the decoupled sampling regions can be problematic. For the decoupled vertical sampling region we determine the maximum climb height of the aircraft, as a function of the focal length F and the dynamic constraints of the aircraft, and set it as the semi-minor axis. The dynamic constraints of an aircraft limit the vertical sampling region, but there are no constraints limiting the horizontal sampling region. To ensure that a failure to determine a path is not due to an underestimated sampling region size, we selected a very large horizontal semi-major axis $b = 2F$. We assume that the implementation of a dynamic sampling region (see Section 5.8.3 for details) will reduce the computational inefficiency caused by the large horizontal sampling region.

5.5.1.2 Uniform Sampling

Now that the sampling region size has been determined, we can generate the randomly sampled states within. The coordinate sampler makes use of a uniform sampling strategy (see Section 4.3). Uniform sampling has

been chosen due to its simple implementation while still retaining a favourable density characteristic (see Section 4.3). The generation of uniform samples within an ellipse is achieved by applying a transformation to uniformly distributed samples generated within a unit circle.

One method to sample within a circle is to make use of uniformly distributed polar coordinates (r, α). This method unfortunately does not produce uniformly distributed samples, and making use of this sampling strategy will result in the clustering of samples around the origin. In order to achieve a truly uniform PDF within a circle we have to weight the radius r towards the perimeter of the circle [77]. The reason for this becomes clear if we look at an arbitrary wedge defined by $(\alpha, d\alpha)$. If we wish to have a uniform distribution more samples have to be taken further from the centre. The weighting of r is accomplished by making use of the $\sqrt{r}$ in the parametric equations:

$$n_c = \begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} x_s \\ z_s \end{bmatrix} = \begin{bmatrix} \sqrt{r}\cos\alpha \\ \sqrt{r}\sin\alpha \end{bmatrix}, \tag{5.6}$$

where

$$r = U[0:1] \tag{5.7}$$

$$\alpha = U[0:2\pi]. \tag{5.8}$$

In Equation 5.6 n_c is a randomly sampled coordinate within the unit circle. Note that the generation of the samples for the y_s and z_s are equivalent. This is due to the decoupling of the system.

The points generated by Equation 5.6 will be uniformly distributed within a unit circle. In order to transform these to an elliptical sampling region we have to apply a transformation matrix that transforms the unit circle to an ellipse. This is accomplished by applying a 2×2 reformation matrix $\mathbf{T}_d$ that is diagonally populated by the appropriate major or minor axis of the ellipse. Finally the elliptical sampling region must be (1) translated by an offset s_q, thereby placing the foci of the ellipse at q_{init} and q_{goal}, and (2) rotated by Ψ_s to ensure that the orientation of the sampling region lies between the q_{init} and q_{goal}. The transformed uniformly sampled state n, which lies within the predefined elliptical sampling region between the initial and goal states, is described by:

$$n = \Psi_s[\mathbf{T}_d n_c + s_q]. \tag{5.9}$$

We made use of Equations 5.6 and 5.9 to generate multiple samples within an elliptical shape.

5.6 Local Planning Method

A local planning method (LPM) is required to determine whether a sample state is dynamically realisable. If it is the LPM determines the dynamically feasible path that connects a vertex in the tree to the newly

sample state. An LPM is essential for the application of a sampling-based path planner to dynamically constrained vehicle. The development of a kinodynamic motion planner requires that the system cater for the dynamic or non-holonomic constraints of the vehicle. The application of a path planning algorithm to an under-actuated, non-holonomic system requires the generation of a viable action space. Both under-actuated and non-holonomic constraints can be derived from the generalised equation of motion for the aircraft [56]:

$$\dot{q} = f(q, u), \tag{5.10}$$

where $q \in \mathcal{C}$ is the state of the aircraft, $\dot{q}$ is the time derivative of q and $u \in \mathcal{U}$. The sets $\mathcal{C}$ and $\mathcal{U}$ define the configuration or state space, and control space of the aircraft, respectively. The system can be defined as under-actuated if the dimensions of $\mathcal{C}$ are larger than $\mathcal{U}$, implying that the system contains fewer control inputs than physical states [56]. To simplify the management of such a system we have to define an action or manoeuvre space, where a manoeuvre defines any controlled change in the vehicle's state that adheres to the differential constraints of that vehicle. By defining an action space we are able to decode the required path into a set of dynamically realisable manoeuvres.

5.6.1 Action Space

The action or manoeuvre space encodes all the relevant dynamic constraints of the vehicle into a lower dimensional state space $\mathcal{M}$ [7]. The manoeuvre space can be defined as:

$$\mathcal{M} = \mathcal{C} \times \mathcal{C}_{\mathcal{M}}, \tag{5.11}$$

where $\mathcal{C}_{\mathcal{M}}$ is a set of all the viable manoeuvres [78]. Each of the manoeuvres in $\mathcal{C}_{\mathcal{M}}$ comprises an action and a corresponding completion time. The manoeuvre space $\mathcal{M}$ in Equation 5.11 is defined as a set of states that are made up of both C-space ($\mathcal{M}_{\mathcal{C}} \subseteq \mathcal{C}$) and manoeuvre ($\mathcal{M}_m \subseteq \mathcal{C}_{\mathcal{M}}$) components. Where $\mathcal{M}_{\mathcal{C}}$ is the C-space components and $\mathcal{M}_{\mathcal{M}}$ is the manoeuvre components of the manoeuvres space. These C-space components represent all the positions and orientations that the aircraft can attain, while the manoeuvres refer to a set of straight, ascend, descend or turning commands. The advantages of implementing an action space are listed below [7].

1. The physical limitations and dynamic constraints of the aircraft are encoded into each manoeuvre, thereby removing the need to check if each path is flyable.
2. The mean and covariance of each path can be propagated off-line and stored, to reduce computation time as on-line propagation is not required.
3. The application of the manoeuvre space to the path planning problem completely abstracts the execution and calculation of each manoeuvre from the path planner, vastly simplifying its complexity.

Unfortunately it is very difficult to determine a set of manoeuvres that can describe the full dynamic capabilities of a vehicle, while still retaining a relatively low dimensional manoeuvre space. Therefore the

implementation of an action space can reduce the complexity of the system, but also reduces the manoeuvrability of the aircraft. This does not necessarily imply that the aircraft's ability to reach the required states is impaired. This makes the selection of effective action space manoeuvres very important to ensure that the simplification does not affect the reachability of the path planner. Sections 5.6.2 and 5.6.3 describe the selection the of practical manoeuvres that will be used to synthesise the horizontal and vertical resolution trajectories for the aircraft.

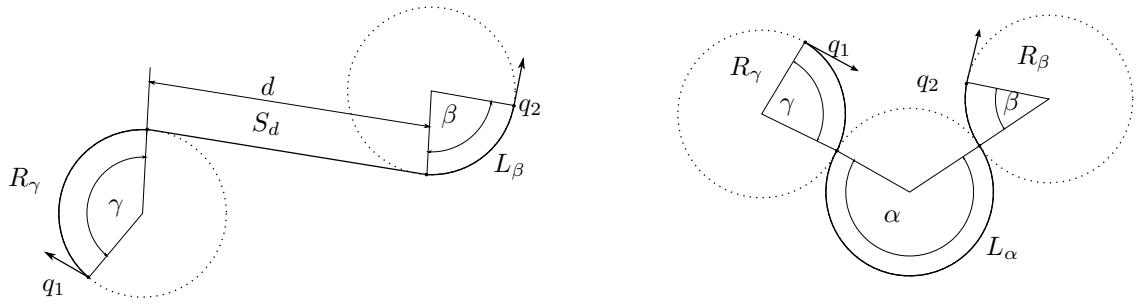
5.6.2 Horizontal LPM

The horizontal domain is defined as the xy-plane, described in Section 5.2, placed at the current altitude of the aircraft. This definition of the horizontal plane requires the selection of initial and goal states at the same altitude to ensure that a horizontal solution is dynamically possible. Within this plane the manoeuvring of an aircraft resembles that of a moving car. Its inputs are comprised of speed (thrust) and turning (bank angle) commands. Generally a yaw control loop is used to give bank angle commands based on the current yaw angle error, but this can result in undesirable transients. Therefore the outer-loop is discarded and a bank angle command is directly used to produce a yaw rate that steers the aircraft to the required heading angle. The thrust command is controlled to ensure a constant speed is maintained throughout the turning manoeuvres.

There exist two promising methods to generate manoeuvres similar to those of a moving car, namely: Dubins curves [79] and Reeds-Shepp curves. A major difference between the two methods is that Reeds-Shepp curves allow for a negative speed (reversing), while Dubins curves determine paths using only positive velocities [10]. Therefore we model the aircraft as a Dubins car and made use of Dubins curves to determine the required path manoeuvres for the horizontal plane. Dubin showed that a path between any two states can be achieved by the Dubins car model using three manoeuvres [79]. Where a manoeuvre describes a single predefined action over a certain time period, a continuous combination of manoeuvres is referred to as a manoeuvre sequence, and a set of these manoeuvre sequences is described as a manoeuvre set. Each of these manoeuvres makes use of a constant reference command (left, right or straight) and an execution time t_m [10]. The resultant Dubins curves are a sequence of circular path segments (the turns) and straight segments as seen in Figure 5.4. The dynamically realisable path between the two states v_1 and v_2 is described by three manoeuvres: a right turn $R_\gamma(t)$, straight segment $\mathrm{S}_d(t)$ and a left turn $L_\beta(t)$ in Figure 5.4a, while Figure 5.4b shows how the two states are connected using three turning manoeuvres: $R_\gamma(t), L_\alpha(t), R_\beta(t)$. Here the angles $(\gamma, \alpha, \beta) \in [0, 2\pi)$ denote the required change in heading angle for the turns, t defines the time required to execute each manoeuvre and the $d \geq 0$ defines the distance travelled by the straight segments.

The transition from a turning to a straight manoeuvre occurs at the tangent point between the straight line and the turning circle. This is required to ensure that the manoeuvre sequence remains geometrically continuous. See Section 5.6.4 for the practical complications of geometrically ideal manoeuvre sequences.

The constant speed assumption required for the implementation of Dubins curves results in a direct proportionality between distance and time. Therefore we can determine the exact time required to execute each manoeuvre from its corresponding distance. We already have the straight segment distance d. All that is required is the calculation of the turning arcs' distances. This can be determined from the required angles (γ, α, β) and the radius of the turning circles.



(a) R_γ,S$_d$, L_β (b) $R_\gamma, L_\alpha, R_\beta$

Figure 5.4: Typical Dubins curve determined between two states q_1 and q_2, adapted from LaValle [10]

Curve Radii

In order to determine the required radii of the turning circles we have look at the aircraft's dynamic constraints in the horizontal plane. The horizontal dynamics of an aircraft are limited by thrust and bank angle constraints. These limitations can be translated to a dynamically constrained minimum turning radius r_{min}, defined as a function of the magnitude of the current velocity $\bar{V}$ and maximum bank angle of the aircraft ϕ_{max}. By selecting the minimum turning radius we attempt to ensure maximum manoeuvrability of the aircraft, while maintaining a constant altitude. The function used to determine r_{min}, assuming no wind disturbance [13] is shown below:

$$r_{min} = \frac{\bar{V}^2}{g \tan \phi_{max}}, \tag{5.12}$$

where g is the Earth's gravitational force at the location of the aircraft. Another method of determining r_{min} is by limiting the angular rate of the aircraft. The radius of the turn is therefore determined by a set maximum rate of turn ω_{max}, as shown in Equation 5.14.

$$\omega_{max} = \frac{\bar{V}}{r_{min}} \tag{5.13}$$

$$r_{min} = \frac{\bar{V}}{\omega_{max}} \tag{5.14}$$

Either of these methods can be implemented to function as the constraint limiter in the LPM module. By using these limited bank angle commands we can determine the required Dubins curves between any two states in the horizontal plane. The two Dubins Curve methods used in the generation of the paths for the conflict detection module proposed in this thesis are discussed below. The application of these methods is defined by the dimension of the Manoeuvre Space in which they are applied. When generating the tree structure for the adapted RRT*, the LPM functionality can be divided into two separate motion planners: one that connects newly sampled states to the tree and another that connects sampled states to the goal state. The differences between the two motion planners are in the composition of the final states, where the final state represents the state to which the LPM connects. When a new sample is added to the tree,

contains only position parameters, and no constraints are placed on the attitude parameters. This is due to the nature of the sampling method used by the adapted RRT* algorithm (see Section 5.5.1 for details). Due to the lack of a goal heading, the required LPM can be simplified to connect the initial state to a set of defined coordinates, with any convenient heading. This simplified LPM implementation is described in Section 5.6.2.1, while the general Dubins curves used to connect to the desired goal state, comprised of a set of desired coordinates as well as a desired heading, are described in Section 5.6.2.2.

5.6.2.1 Two-Manoeuvre Horizontal LPM for Connecting to Intermediate States

The motion planner described in this section is required to connect a state (position and heading) q to a position s, where q is a vertex in the tree and s is the position of the newly sampled state. This reduces the complexity of the motion planning problem by removing the heading condition. This simplified motion planner can be implemented using a two dimensional (2D) manoeuvre space $\mathcal{M}^2$, implying that in order to successfully connect q and s we require only two manoeuvres: a turning segment and a straight segment. This reduced manoeuvre set is depicted in Table 5.1. The reduction in the manoeuvre set is visible when looking at Table 5.2. A set of these manoeuvres is required to implement the general Dubins Curves in $\mathcal{M}^3$.

Table 5.1: The simplified manoeuvre set required to connect any two states (excluding the final states heading) in the two dimensional (2D) horizontal space, depicting the 2 possible manoeuvre sequences

	Manoeuvres	
Sequence	Type	
1	$R_\gamma(t_1)$	$\mathrm{S}_d(t_2)$
2	$L_\gamma(t_1)$	$\mathrm{S}_d(t_2)$

The 2D Dubins curve, as seen in Figure 5.5, makes use of a single turning and straight manoeuvre to connect q and s. The LPM always makes use of an initial turning manoeuvre. Each manoeuvre is defined as either right, left or straight (R, L or S), as seen in Table 5.1 and is executed for t seconds. Once the path to s has been determined we can derive the heading at this coordinate from the calculated path. Therefore the velocity vector at s is derived from the manoeuvres generated by the LPM. Using this, the entire sampled state (position and attitude parameters) is determined.

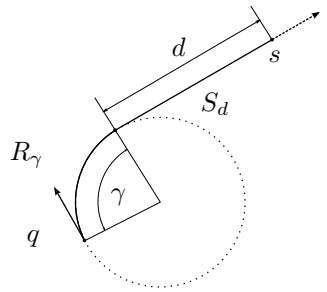
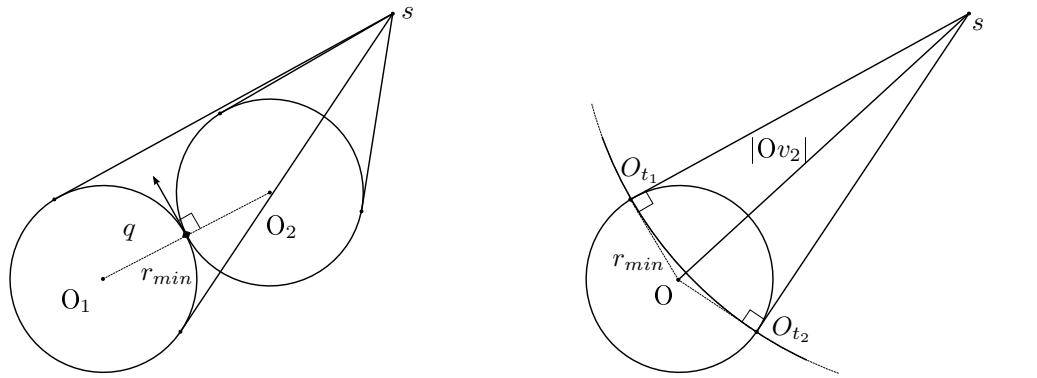


Figure 5.5: Two manouevre Dubins curve determined between two states v_1 and v_2

Path Calculation

The calculation of the required manoeuvre sequence between an arbitrary state q and position s, starts by first determining the two turning circles. This is accomplished by placing two circles, centred at O_1 and O_2 with a radius of r_{min}, perpendicular to the initial state velocity vector, as seen in Figure 5.6a. These two circles represent all the possible turns, at the set bank angle, that can be executed to reach the desired coordinate s. After the generation of the turning circles we have to determine all the required straight segments. These straight segments can be determined by finding all the straight lines that are tangential to each circle and pass through the desired final coordinate. This is geometrically represented in Figure 5.6b when regarding a single circle. The tangent points (O_{t_1},O_{t_2}) are calculated by first placing a circle at the coordinate s with a radius of |O,s|. From this it is possible to determine the two intersection points between the circles at O and s. These points correspond to the required tangent points.



(a) Placing two circles perpendicular to the initial velocity vector at q

(b) Tangent calculation between an arbitrary circle at O, with a radius of r_{min}, and a point s

Figure 5.6: Typical Dubins curve determined between a state q and coordinate s, adapted from LaValle [10]

The calculation of all the manoeuvre sequences between the two states yields 4 possible solutions. Of these, only two are dynamically realisable. The incorrect manoeuvre sequences are removed by comparing the rotational vectors. In order for a sequence of manoeuvres to be dynamically realisable. its rotational vector has to be the same as the current rotational vector of the initial state. These rotational vectors are determined by making use of the cross product rule:

$$\mathbf{a} \times \mathbf{b} = |\mathbf{a}||\mathbf{b}| \sin(\alpha), \tag{5.15}$$

where $\mathbf{a}, \mathbf{b}$ represent arbitrary vectors and α is the angle between them. We determine the rotational vector of each manoeuvre sequence and compare them to the rotational vector of the initial state about each circle, thereby identifying the dynamically realisable paths and removing the others. Before calculating the possible resolution paths a test is conducted to determine whether s lies on either O_1 or O_2. If true, the rotation is determined and the final straight distance is set to zero.

After determining the dynamically realisable paths from q to s we select the shortest path and determine the required angle γ and corresponding execution times t_1, t_2. This results in a dynamically flyable path segment

from q to s that can be described using one of the manoeuvre sequences in Table 5.1. Finally we calculate the heading angle for the final state s from the selected manoeuvre sequence.

5.6.2.2 Three-Manoeuvre Horizontal LPM for Connecting to the Goal State

The 2D Dubins curve is viable as a methods to connect sampled states when populating the search tree, but in order to accurately connect to the goal state of the system we require a more complex method. This is due to the added heading parameter present at the goal state. In order to ensure that the goal state is reached perfectly (position and heading) we require the more advanced LPM method. The general Dubins curve method is capable of achieving this by connecting two arbitrary states: q_1 and q_2 perfectly, where $q_2 \triangleq q_{goal}$, using three manoeuvres. This method, illustrated by Figure 5.4, is more complex and computationally expensive, but is only executed when a goal state connection is attempted. As described in Section 5.6.2 we require a more complex manoeuvre space to ensure that both the coordinates and heading of the goal state are accurately achieved. The increased manoeuvre space results in a larger manoeuvre set to accurately represent all the possible paths that connect any two states, as seen in Table 5.2.

Table 5.2: The manoeuvre set required to connect any two states exactly in a 2D horizontal space, depicting the 6 possible manoeuvre sequences, adapted from LaValle [10] and Cowley [14]

	Manoeuvres		
Sequence	Type		
1	$R_\gamma(t_1)$	$\mathrm{S}(t_2)$	$L_\beta(t_3)$
2	$R_\gamma(t_1)$	$\mathrm{S}(t_2)$	$R_\beta(t_3)$
3	$L_\gamma(t_1)$	$\mathrm{S}(t_2)$	$L_\beta(t_3)$
4	$L_\gamma(t_1)$	$\mathrm{S}(t_2)$	$R_\beta(t_3)$
5	$R_\gamma(t_1)$	$L_\alpha(t_2)$	$R_\beta(t_3)$
6	$L_\gamma(t_1)$	$R_\alpha(t_2)$	$L_\beta(t_3)$

The calculation of the required circular turns and tangential straight path segments becomes more complex due to the additional dimension.

Path Calculation

The calculation of the general Dubins curve manoeuvres requires the grouping of the possible sequences, described in Table 5.2. Two groups are defined based on the second manoeuvre in the sequence. One group contains the straight segment manoeuvres and the other a turn. These groups can therefore be defined as either $[TST]$ or $[TTT]$, where T refers to a turn manoeuvre and S depicts a straight manoeuvre.

The generation of the $[TST]$ manoeuvre sequence is similar to the method described in Section 5.6.2.1. We start by placing four circles as seen in Figure 5.7 perpendicular to the initial and goal states' respective velocity vectors. The dotted lines, in Figure 5.7, between the circles represent all possible straight line connections that have to be tested. In Section 5.6.2.1 there are 4 straight manoeuvre connections, while for the general Dubins curve implementation the possible straight manoeuvre connections amount to 16. The increased dimension size results in an exponential increase in the calculations, excluding the additional $[TTT]$ computation that is also required.

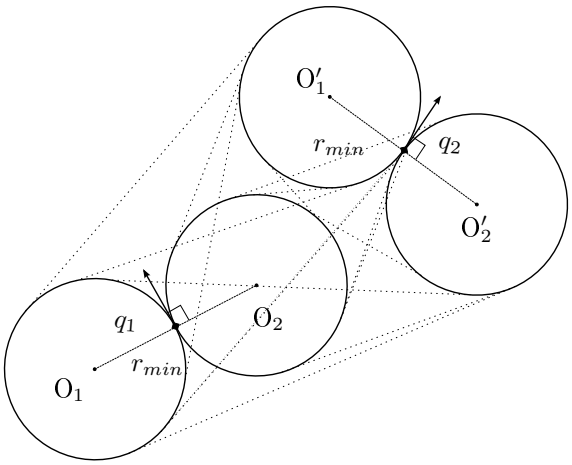


Figure 5.7: Placing four circles perpendicular to the velocity vecotors of q_1 and q_2

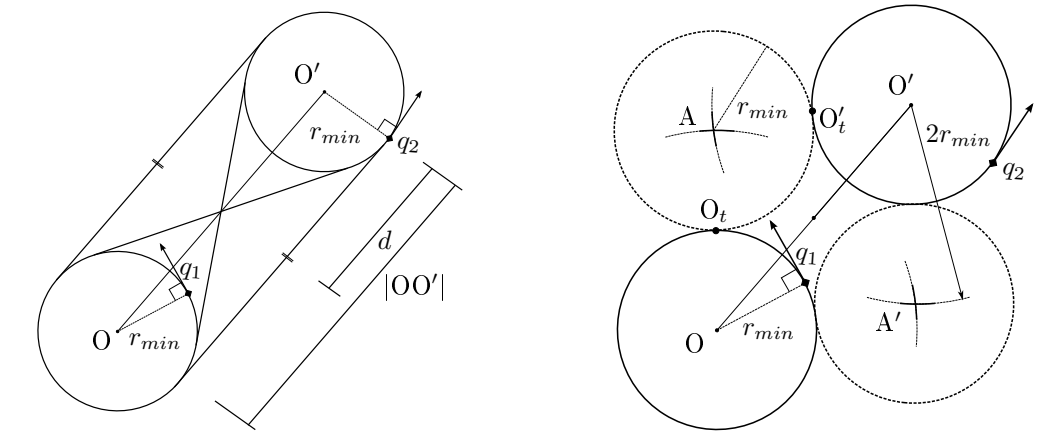
All the possible smooth connection points between the straight and turning manoeuvres translate to the tangent points between each circle pair. Figure 5.8a graphically represents an arbitrary congruent[1] circle pair centred at O and O′. In order to calculate the tangent lines we have to identify some characteristics that arise when dealing with a congruent circle pair. The tangent line pairs can be divided into the outer tangent lines and the inner tangent lines. The point of intersection between the inner tangent lines is known as the internal similitude centre, while the point of intersection between the outer tangent lines is the outer similitude centre. When dealing with congruent circles there exists no outer similitude centre, because the outer tangent lines are parallel to each other as well as to the line OO′. Therefore the intersection point of the outer tangent lines corresponds to the points perpendicular OO′ on the edge of the circles.

The inner tangent lines' intersection points can be determined by making use of the inner similitude centre. The inner similitude centre for congruent circles corresponds to the halfway point of OO′. Therefore we have to determine tangent lines between each circle and the inner similitude centre. This corresponds to the calculation of the tangent lines between each circle and a point, as in Section 5.6.2.1. Using those methods we can determine all the tangent lines and their intersection points. The flyable straight manoeuvres are identified by comparing the rotations of the tangents, as in Section 5.6.2.1.

The calculation of the $[TTT]$ manoeuvre sets, as seen in Figure 5.8b, is enabled when $|\mathrm{OO}'| \leq 4r_{min}$. This ensures that the tangent circles A,A′ can exist. If the initial and goal states are within range the circles with equivalent rotational vectors are selected. In order to implement a $[TTT]$ manoeuvre sequence, each turn has to have an opposite rotational vector, about their centre, to ensure a dynamically realisable path (see Figure 5.4b). Once the centres of these tangent circle pairs have been determined we can calculate the intersection points (O_t,O'_t). Thereafter the rotation of the tangent circle is set to the opposite of initial and goal circles. Finally we determine the arc angle between the O_t and O'_t based on the set rotation of the circle.

Now that the entire manoeuvre set between the two states has been determined, we select the shortest path as the desired manoeuvre sequence to be implemented. The mathematical calculations required to determine the tangent points that intersect the turning circles are supplied in Appendix B.

[1] Congruent circles are any circles that have the same radius, area and circumference.



(a) The geometric representation required to determine all possible tangent lines between two circles centered at O and O′, with a radius of r_{min}

(b) Determine the tangent circles' centers (A, A′) from the intersection between two circles with a radius of $2r_{min}$ placed at O and O′

Figure 5.8: Typical Dubins curve determined between two states v_1 and v_2, adapted from LaValle [10]

5.6.3 Vertical LPM

The vertical domain is defined as the xz-plane at the current horizontal position of the aircraft. The vertical resolution LPM ensures that the dynamic constraints pertaining to all vertical manoeuvres are adhered to. These translate to constraints on the climb rate $\dot{h}$ and flight path angle θ, discussed in Section 3.1.4.

The implementation of a vertical LPM module requires the development of a set of possible climb rates and flight path angles. This set must ensure that all the constraints that pertain to vertical manoeuvres are not violated. In order to simplify development of these manoeuvre sets we have made a few assumptions. As with the horizontal LPM (see Section 5.6.2) we assume that the velocity of the aircraft remains constant for the entire resolution process. This ensures that we can predict the final state's arrival time accurately, from the connecting manoeuvre sequence. An additional assumption is made that the velocity vector at each state represents straight and level flight, as seen in Figure 5.9. This means that all vertical LPM calculations are determined from a straight and level flight position. Alternatively it implies that we assume that the aircraft is dynamically capable of inverting its current flight path angle θ, therefore allowing the system to command a descent while an ascent is in progress.

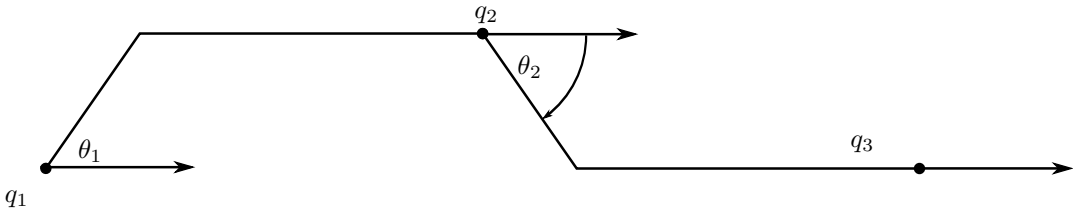


Figure 5.9: Vertical manoeuvre squences connecting states q_1, q_2 and q_3

These assumptions allow the development of manoeuvre sets that trace the path shown in Figure 5.9, where θ represents the climb or descent angle required. The LPM aims to connect the two states q_1 and q_2 by making

use of an initial climb or descent manoeuvre, followed by a straight and level flight. This implementation of the LPM results in the manoeuvre set depicted in Table 5.3, where $A_\theta(t)$ refers to the altitude shift at a flight path angle of θ.

Table 5.3: The manoeuvre set used to connect two states in the 2D vertical space

Manoeuvres		
Sequence	Type	
1	$A_\theta(t_1)$	$\mathrm{S}_d(t_2)$
2	$-A_\theta(t_1)$	$\mathrm{S}_d(t_2)$

Climb Rate Selection

The selection of realistic climb rates is of paramount importance to the effectiveness of the resolution module as well as the LPM. The selection of conservative climb rates could negatively affect the resolution algorithm's ability to find a path. Alternatively large climb rates could affect the pilot's or autopilot's ability to follow the path and additionally negate the assumptions required to implement the vertical LPM. We selected two sets of climb and descent rates based on an approximation of the current TCAS climb and steep climb advisories (see Section 2.2.2 for details). These translate to a basic climb and descent rate of 1500 ft/min and a steep climb and descent rate of 2500 ft/min.

Path Calculations

The calculation of the vertical manoeuvre set makes use of the climb and descent rates defined above to determine the required manoeuvre sequence. This is accomplished by determining the required climb or descent angle θ between the states q_1 and q_2 by making use of Equation 5.16.

$$\theta = \arcsin \frac{\dot{h}}{\bar{V}} \tag{5.16}$$

Using this angle in conjunction with the change in altitude we can determine the distance travelled during the altitude manoeuvre. Alternatively we can determine the time required to implement the climb manoeuvre by making use of:

$$t_{A_\theta} = \frac{\Delta z}{\dot{h}}, \tag{5.17}$$

where Δz refers to the change in altitude between q_1 and q_2. From either method we can derive the required time and distance of A_θ as well as the coordinates after the execution of the manoeuvre. Using the intermediate coordinate (after the initial climb or descent) and the final state we can derive the straight and level manoeuvre required to connect them. The final manoeuvre sequences can connect two vertically separated states using a dynamically realisable path (assuming the selection of accurate climb rates), if one exists.

5.6.4 Practical Implications

The horizontal and vertical LPM functions described above generate geometrically perfect manoeuvre sequences between two desired states. The use of geometrically ideal manoeuvres vastly simplifies the LPM module, since the manoeuvre sets can be determined from predefined geometric formulas. This simple implementation of the LPM makes it a computationally attractive motion planning method.

The paths determined by these sequences are geometrically ideal and therefore it is practically impossible for the aircraft to exactly traverse the path. However, Dubin showed that the manoeuvre sequences are feasible in terms of reachability [79]. Assuming efficient cross track and altitude control (see Section 3.1.2) we can expect an aircraft to reach the desired trajectory. Therefore accurate steady state (after a long time) tracking of the horizontal manoeuvres can be assumed, while the altitude shifts A_θ are expected to be tracked with an offset. Additionally, deviations from the path are to be expected at the transition points, resulting in an overshoot of the predicted path segment. However, these transients and offsets cause negligibly small deviations and can be neglected except for conflict prediction. Therefore the algorithm assumes that a simulation model (see Section 5.4) will be implemented by the conflict detection system to accurately determine the mean path traversed by the aircraft. The transients expected at the transition points as well as any offsets should be encapsulated by the simulation module's path prediction, thereby reducing the uncertainty of the aircraft state about these points. Ultimately we expect all states to be reached accurately and that the states can be reached by a dynamically realisable path. We do concede that the exact path flown may not perfectly coincide with the geometrically perfect manoeuvre sequence; however, this should not affect the integrity of the system given the implementation of an accurate simulation model.

5.7 Path Optimisation

The implementation of an optimal conflict resolution algorithm requires some definition of what can be considered an optimal solution. The definition of an optimal solution can be expressed by minimising a cost function C, that can be represented by:

$$C(m) \triangleq \int_{t_0}^{t_1} f(m(\delta))\mathrm{d}\delta \tag{5.18}$$

where δ is the integration variable and $m(\cdot)$ represents the desired manoeuvre applied over the time interval $[t_0, t_1]$ [7]. Using this cost function it is possible to determine the cost of the entire manoeuvre sequence. The cumulative cost function C can be used to determine the cost of each manoeuvre and summed to determine the manoeuvre sequence cost. The formulation of f determines the optimisation criteria. Numerous different methods have been proposed to construct the cost function based on time, energy, path length and fuel consumption [7, 80]. Each method defines a unique optimisation criterion and thereby promotes the construction of different paths. Some cost functions include the probability of conflict as an optimisation criterion, when implemented with a probabilistic conflict detector [81]. In order for a conflict resolution system to function reliably the probability of conflict must be a non-negotiable safety requirement [7]. This

is achieved by applying a fixed threshold, which determines whether a path is indeed conflict-free. Therefore a cost function is only applied to paths deemed conflict-free.

We have identified three cost functions that have been tested using the adapted RRT* algorithm. These include distance, energy and time integral of path deviation (path deviation), explained in Sections 5.7.1, 5.7.2 and 5.7.3. The expected behaviour and optimal solutions of the selected cost functions are shortly discussed in Section 6.2.1.

5.7.1 Distance Cost Function

The distance-based cost function is based on the Euclidean distance of the path, therefore the system will attempt to determine the shortest path to the goal state when applying this cost function. The implementation of the distance-based cost function is simplified by the constant speed assumption, resulting in a direct relationship between time and distance. This relationship implies that each manoeuvre's distance can be determined from its execution time and the constant speed of the aircraft, where execution time is defined from the starting time t_0 to the ending time t_1 of each manoeuvre. The relationship is given by:

$$C_{dist} = \int_{t_0}^{t_1} f(\bar{V}\delta)\mathrm{d}\delta = \bar{V}\Delta t, \tag{5.19}$$

where $\Delta t = t_1 - t_0$. Once the cost of each manoeuvre is calculated, due to the additive nature of the cost function, we can determine the total cost of the manoeuvre sequence as the summation of the individual manoeuvre costs.

5.7.2 Energy Cost Function

The energy-based additive cost function makes use of the energy calculations for an aircraft, using equations provided by Anderson [82] and Yechout [11], to determine the costs of each manoeuvre. In order to calculate the energy required to execute each manoeuvre sequence, we have to determine the required power for each manoeuvre. Therefore each unique manoeuvre is identified and their corresponding power requirement is determined. The different manoeuvres are identified below along with their corresponding power calculations.

5.7.2.1 Power of Straight and level flight

The straight manoeuvre S(t) can be described as straight and level flight, where straight and level flight implies that the aircraft is operating in complete equilibrium. This means that all the forces are balanced and all rates (heading and climb) are zero. Mathematically a flight segment is defined as being straight and level when

$$\mathrm{T} = \mathrm{D} \tag{5.20}$$

$$\mathrm{L} = \mathrm{W}, \tag{5.21}$$

where the forces are thrust (T), drag (D), lift (L) and weight (W). The lift and drag forces can be modelled using equations containing the aerodynamic coefficients of the aircraft. These coefficients describe the aerodynamic characteristics of a specific aircraft and can be considered independent of the flying speed of the aircraft [5]. The two coefficients that influence the calculation of the lift and drag forces are C_D and C_L, the drag and lift coefficients, respectively. The relationship between the forces and their corresponding coefficients is described by:

$$\mathrm{L} = \frac{1}{2}\rho\bar{V}^2 S C_L \tag{5.22}$$

$$\mathrm{D} = \frac{1}{2}\rho\bar{V}^2 S C_D, \tag{5.23}$$

where ρ is the current air pressure and S is the area of the wing. A relationship between the drag and lift coefficients, shown in Equation 5.24, can be calculated based on the physical characteristics of the aircraft as well as the environmental conditions.

$$C_D = C_{D_0} + \frac{C_L^2}{\pi e A R} \tag{5.24}$$

In Equation 5.24, C_{D_0} is the parasitic drag, e is the Oswald efficiency factor and AR is the aspect ratio of the wing (length squared over wing area). The parasitic drag is a combination of the form drag, skin friction and interference drag. The Oswald efficiency factor is determined empirically through wind-tunnel tests and represents the distribution and variation of pressure drag with lift [11]. From Equations 5.21, 5.22, 5.23 and 5.24 it is possible to derive an equation for the drag induced during straight and level flight in the form of [11, 82]:

$$\mathrm{D} = \frac{1}{2}\rho\bar{V}^2 S C_{D_0} + \frac{\mathrm{W}^2}{\frac{1}{2}\rho\bar{V}^2 S}\left(\frac{1}{\pi e A}\right) \tag{5.25}$$

By making use of Equations 5.20 and 5.25 we can determine the required thrust to ensure that the aircraft maintains straight and level flight. The definition of the power required to execute an aircraft manoeuvre can be described as a function of the required thrust and current speed, as seen in Equation 5.26.

$$\mathrm{P}_{req} = \mathrm{T}_{req}\bar{V} \tag{5.26}$$

$$= \mathrm{D}\bar{V} \tag{5.27}$$

By substituting the thrust for the induced drag at $\bar{V}$, we can determine the required thrust from Equation 5.25. This yields the straight and level power equation:

$$\mathrm{P}_{s\&l} = \frac{1}{2}\rho\bar{V}^3 S C_{D_0} + \frac{\mathrm{W}^2}{\frac{1}{2}\rho\bar{V}S}\left(\frac{1}{\pi e A}\right) \tag{5.28}$$

at a specific constant speed $\bar{V}$ and air pressure ρ. This equation enables the calculation of the power required for any straight and level flight segment.

5.7.2.2 Power of Constant Climb or Descent

An increase in the power above the straight and level power requirement will result in acceleration. This acceleration can represent either an increase in the aircraft kinetic or potential energy. Figure 5.10 represents the forces during a climbing manoeuvre.

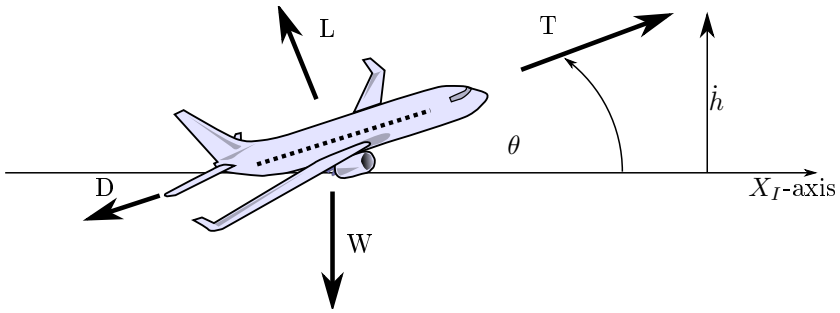


Figure 5.10: Forces for an aircraft during a climbing manoeuvre, adapted from Yechout [11]

From Figure 5.10 it is possible to derive the balanced force equations as a function of the flight path angle θ:

$$\mathrm{L} - \mathrm{W}\cos\theta = \frac{\mathrm{W}}{g}\bar{V}\frac{\mathrm{d}\theta}{\mathrm{d}t} \tag{5.29}$$

$$\mathrm{T} - \mathrm{D} - \mathrm{W}\sin\theta = \frac{\mathrm{W}}{g}\frac{\mathrm{d}\bar{V}}{\mathrm{d}t}, \tag{5.30}$$

where $\bar{V}\frac{\mathrm{d}\theta}{\mathrm{d}t}$ is the normal acceleration and $\frac{\mathrm{d}\bar{V}}{\mathrm{d}t}$ is the acceleration tangential to the flight path of the aircraft. Any vertical change executed by an aircraft can be expressed in terms of a rate of climb or climb rate $\dot{h}$:

$$\dot{h} = \frac{\mathrm{d}h}{\mathrm{d}t} = \bar{V}\sin\theta. \tag{5.31}$$

From Equations 5.30 and 5.31 we can derive a general flight path angle independent equation for the rate of climb:

$$\dot{h} = \bar{V}\left(\frac{\mathrm{T}-\mathrm{D}}{\mathrm{W}}\right) - \frac{\bar{V}}{g}\frac{\mathrm{d}\bar{V}}{\mathrm{d}t} \tag{5.32}$$

Equation 5.32 is further simplified by applying the constant speed assumption. The assumption implies no change in speed and therefore the second term of Equation 5.32 becomes zero, $\frac{V}{g}\frac{\mathrm{d}V}{\mathrm{d}t} = 0$. Additionally this assumption results in a constant rate of climb. Finally we can determine the power required to implement a constant climb rate. The required power can be determine from Equations 5.26, 5.27 and 5.32:

$$\dot{h} = \bar{V}\left(\frac{\mathrm{T} - \mathrm{D}}{\mathrm{W}}\right) \tag{5.33}$$

$$= \frac{P_{avail} - P_{req}}{\mathrm{W}} \tag{5.34}$$

From Equation 5.34 we can determine the additional power required to execute a constant rate of climb:

$$\mathrm{P}_{\dot{h}} = \dot{h}\mathrm{W} \tag{5.35}$$

The total power required to execute a constant climb is determined as the sum of the straight and level power $\mathrm{P}_{s\&l}$ and the required constant climb power $\mathrm{P}_{\dot{h}}$. Alternatively the power of a constant descent can be calculated as the difference between $\mathrm{P}_{s\&l}$ and $\mathrm{P}_{\dot{h}}$. Therefore we make use of the constant climb power to determine the power required for both ascent and descent manoeuvres

5.7.2.3 Power of Constant Turn

The constant turn manoeuvre is defined by a set turning radius based on the current speed and the dynamic constraints of the aircraft. In order to execute this manoeuvre, we have to apply a bank angle command. Additionally we have to increase the angle of attack and thrust to ensure that the aircraft maintains a constant speed and altitude. Failure to apply the additional thrust and angle of attack commands will result in a loss of altitude or stall the aircraft. The increased thrust command will result in an increase in the power requirements of the aircraft. The aircraft shown in Figure 5.11 is performing a constant level turn. In order

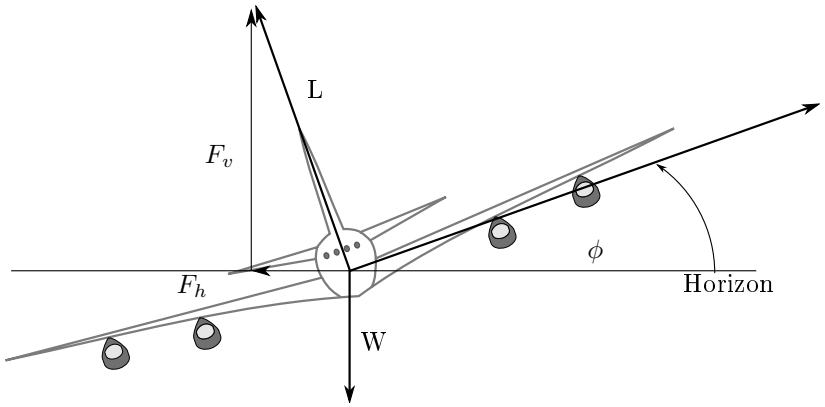


Figure 5.11: Forces actiong on an aircraft during a constant level turn manoeuvre, adapted from Yechout [11]

to ensure a constant altitude, the velocity vector of the aircraft must act only in the horizontal plane [11]. Therefore, as with straight and level flight, the total vertical forces F on the aircraft must remain zero, as in Equation 5.37.

$$\sum F_v = \mathrm{L}\cos\phi - \mathrm{W} = 0 \tag{5.36}$$

$$\therefore \mathrm{W} = \mathrm{L}\cos\phi \tag{5.37}$$

These equations lead us to defined the term load-factor η, derived from Equation 3.1 in Section 3.1.4.

$$\eta = \frac{\mathrm{L}}{\mathrm{W}} = \frac{1}{\cos\phi} \tag{5.38}$$

The load factor of an aircraft is generally defined in terms of the g-force (gs) that the aircraft is pulling [11]. The relationship between the load factor of a generic aircraft and the bank angle, while maintaining a level turn, is shown in Equation 5.38. Now we determine the sum of the forces in the horizontal plane. These forces can be determined using the centripetal force, about a circle with a constant radius, and Pythagorean principles applied to the forces in Figure 5.11. Resulting in:

$$\sum F_h = \frac{m\bar{V}^2}{r} = \frac{\mathrm{W}\bar{V}^2}{gr} = \sqrt{\mathrm{L}^2 - \mathrm{W}^2} = \mathrm{W}\sqrt{\eta^2 - 1} \tag{5.39}$$

$$\frac{\bar{V}^2}{gr} = \sqrt{\eta^2 - 1}, \tag{5.40}$$

where m is the mass of the aircraft. From these equations we can determine the load factor of the aircraft at a specific speed $\bar{V}$ and turning radius r, as seen in Equation 5.42.

$$r = \frac{\bar{V}^2}{g\sqrt{\eta^2 - 1}} \tag{5.41}$$

$$\therefore \eta = \sqrt{\left(\frac{\bar{V}^2}{gr}\right)^2 + 1} \tag{5.42}$$

From the load factor determined above we can derive the total power required to execute the turn. Filippone states that the power required for a constant banked turn grows with a factor of $\eta^{3/2}$ given a constant weight [83]. The relationship, in Equation 5.43, can be used to determine the power required in a turn as a function of the straight and level power of the aircraft and the manoeuvre induced load factor.

$$\mathrm{P}_{turn} = \eta^{3/2}\mathrm{P}_{s\&l} \tag{5.43}$$

5.7.2.4 Energy Calculations

The power calculations described above determine the power required to execute each of the manoeuvres used to implement a complete manoeuvre sequence. The power of each manoeuvre can be used in conjunction with the manoeuvre execution time to determine the energy required to execute each manoeuvre sequence, using:

$$\mathrm{E} = \mathrm{P}t. \tag{5.44}$$

By determining the energy required for each manoeuvre we are capable of selecting the manoeuvre sequences that expend a minimal amount of energy between the initial and final states.

5.7.3 Path Deviation Cost Function

The distance and energy cost functions are both affected by flight time and therefore the distance of the path, resulting in similar attributes between the optimal paths. The path deviation-based cost function is an attempt to implement an alternative method that does not have a large correlation to the path distance. The path deviation-based cost function aims to remain on the current path as long as possible. This is achieved by minimising the time integral of the perpendicular path deviation. Thereby aiming to determine a path the smallest perpendicular deviation from the nominal path, for the shortest time. To the author's knowledge this method is unique in its implementation in the context of a path planning cost function. The method applies a cost to the time deviated from the nominal path (straight) between the initial and goal states. This nominal path is defined at zero perpendicular deviation, therefore it lies on the $d_\perp = 0$ axis. The cost function determines the path deviation using:

$$C_{dev} = \int_{t_0}^{t_1} f(d_\perp, \delta)\ \mathrm{d}\delta, \tag{5.45}$$

where $f(\cdot)$ is a function of the perpendicular distance ($d_\perp$) from the straight path and time. The perpendicular distance is defined as the distance deviated from the original path.

The method makes use of the different manoeuvres and determines their equivalent perpendicular distance function. This is used in conjunction with the execution times to determine the accumulative path deviation-based cost for each manoeuvre sequence.

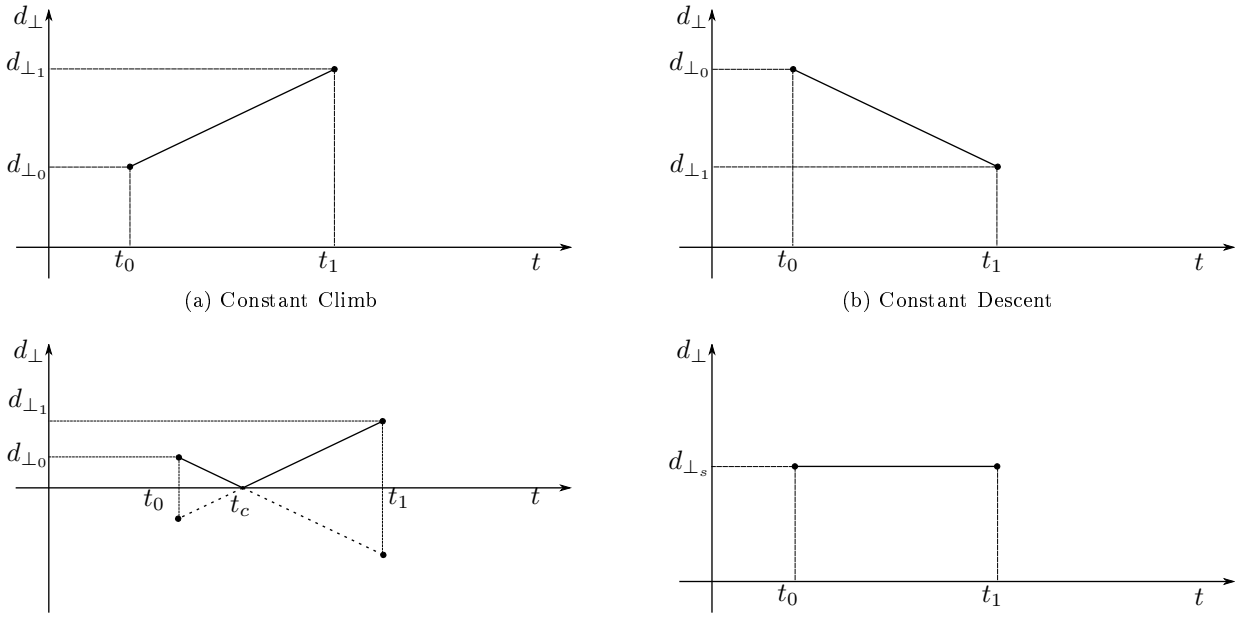
5.7.3.1 Vertical Cost Function Calculation

The path deviation-based cost function applied to the vertical domain is described by:

$$f(d_\perp, \delta) = \left| \frac{\Delta d_\perp}{\Delta t}\delta + d_{\perp_c} \right|, \tag{5.46}$$

where $\Delta d_\perp = d_{\perp_1} - d_{\perp_0}$, $\Delta t = t_1 - t_0$ and $d_{\perp_C}$ is the perpendicular distance at $t = 0$. This can be represented by the four figures shown in Figure 5.12. All manoeuvres in the negative plane are reflected into the positive domain. Each manoeuvre shown in Figure 5.12 has a starting and final deviation $(d_{\perp_0}, d_{\perp_1})$ executed over the time interval of $[t_0, t_1]$.

Figure 5.12c intersects the $d_\perp$-axis at a crossing time denoted by t_c, while the straight flight segment's deviation is denoted by $d_{\perp_s} = d_{\perp_1} = d_{\perp_0}$. The dotted lines, in Figure 5.12c, depict the trajectories of the climb or descent manoeuvres that have to be reflected into the positive domain, since $d_\perp$ has to remain positive.



(a) Constant Climb

(b) Constant Descent

(c) Constant or Descent across the straight path

(d) Straight and level flight

Figure 5.12: Representative path deviation-based function for the different vertical and horizontal manoeuvres

The path deviation of the different vertical manoeuvres shown in Figure 5.12 is equivalent to the area under each manoeuvre. By shifting the time axis by t_0 we can place the start of each manoeuvre at time = 0. This results in Equation 5.47:

$$C_{dev} = \int_0^{\Delta t} f(d_\perp, \delta)\ \mathrm{d}\delta = \int_0^{\Delta t} \left| \frac{\Delta d_\perp}{\Delta t}\delta + d_{\perp_0} \right| \ \mathrm{d}\delta. \tag{5.47}$$

This equation allows the calculation of the path deviation for the three scenarios shown in Figures 5.12a, 5.12b and 5.12d. An alternative calculation is required for manoeuvres that cross the $d_\perp = 0$ axis.

The path deviation of the manoeuvre in Figure 5.12c is represented by the area under the solid line. Therefore we have to determine the area under both triangles. This is achieved by manipulating the $\Delta d_\perp$ and Δt of Equation 5.47 and applying them to both triangles in Figure 5.12c as seen in:

$$C_{dev} = \int_0^{\Delta t_1} \frac{\Delta d_{\perp_1}}{\Delta t_1}\delta + d_{\perp_0}\ \mathrm{d}\delta + \int_0^{\Delta t_2} \frac{\Delta d_{\perp_2}}{\Delta t_2}\delta\ \mathrm{d}\delta, \tag{5.48}$$

where

$$\Delta t_1 = t_c - t_0 \tag{5.49}$$

$$\Delta t_2 = t_1 - t_c \tag{5.50}$$

$$\Delta d_{\perp_1} = d_{\perp_c} - d_{\perp_0} = -d_{\perp_0} \tag{5.51}$$

$$\Delta d_{\perp_2} = d_{\perp_1} - d_{\perp_c} = d_{\perp_1}. \tag{5.52}$$

Equations 5.51 and 5.52 can be reduced because $d_{\perp_c} = 0$ will always hold true. We note from Figure 5.12c that Equation 5.48 is valid for both climb and descent manoeuvres, crossing the $d_\perp$ axis. Using Equations 5.47 and 5.48 we are capable of determining the time integral of path deviation for all of the possible vertical manoeuvres. Each individual manoeuvre's path deviation cost can be summed to determine the cost of an entire manoeuvre sequence.

5.7.3.2 Horizontal Cost Function Calculation

The calculation of the horizontal path deviation results in a total of 7 different manoeuvre transformations. Due to the nature of the horizontal manoeuvres all the possible deviations shown in Figure 5.12 exist, as well those seen in Figure 5.14. An additional calculation applies to the turning manoeuvres. Their derivation requires the application of the transformation function in Equation 5.53, resulting in the transformation seen below:

$$f_H(d_\perp, \delta) = |r\sin\omega\delta + d_{\perp_O}| \ \ |_{\delta_0}^{\delta_1}, \tag{5.53}$$

where r represents the radius of the turn, ω denotes the rate of turn defined as $\frac{\text{radians}}{\delta_1 - \delta_0}$, δ depicts the execution time within the interval $[\delta_0, \delta_1]$ and $d_{\perp_O}$ is the perpendicular distance to the centre of the circle.

The path deviation cost can therefore be determined as the area under the sinusoidal segment between t_0 and t_1, as seen in Equation 5.54:

$$C_{dev} = \int_{t_0}^{t_1 s} f_H(d_\perp, \delta)\ \mathrm{d}\delta = \int_{t_0}^{t_1} |r\sin\omega\delta + d_{\perp_O}|\ \mathrm{d}\delta. \tag{5.54}$$

This integral can be complex to derive, therefore we require a simplified method. We have identified the three different manoeuvre conditions that affect this area in Figure 5.14. By accurately identifying and solving

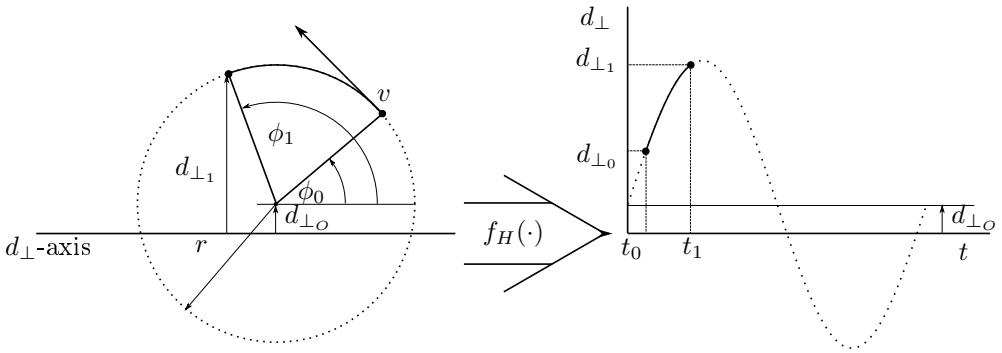


Figure 5.13: Transformation of a circular turn to the perpendicular deviation versus time domain, where $\phi = \omega\delta$

these manoeuvres we can derive a simple solution for Equation 5.53 that can be applied to any of the possible horizontal manoeuvres.

Figures 5.14a and 5.14b represent all the manoeuvres that do not intersect with the $d_\perp$–axis. Individually Figure 5.14a represents any manoeuvre that occurs in the positive plane, while Figure 5.14b occurs in the negative plane (dotted line). Finally Figure 5.14c depicts all manoeuvres that cross the $d_\perp$–axis. The cost of these manoeuvres is generally more complex and require additional computation. Note that the formulation of Equation 5.53 requires that the absolute value of each sinusoidal manoeuvre must be determined, therefore all manoeuvres in the negative plane (dotted lines), shown in Figures 5.14b and 5.14c, are reflected into the positive domain. If a simple solution to each of the individual manoeuvre conditions, described above, can be determined, coupled with their activation criteria, we can solve for each possible solution of Equation 5.53.

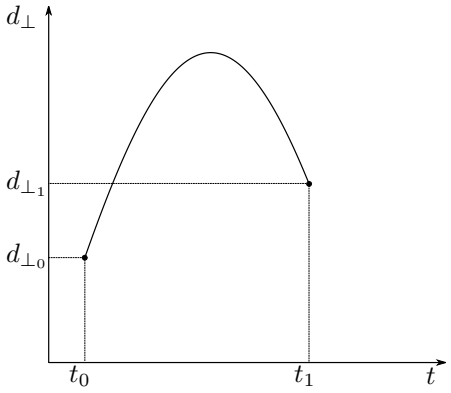
The cost of manoeuvres or manoeuvre segments that do not cross the $d_\perp$–axis, defined by $r\sin\omega\delta + d_{\perp_0} \neq 0$ for all δ within $[\delta_0, \delta_1]$, can be determined using Equation 5.55.

$$\int_{\delta_0}^{\delta_1} |r\sin\omega\delta + d_{\perp_O}|\ \mathrm{d}\delta = \begin{cases} \int_{\delta_0}^{\delta_1} r\sin\omega\delta + d_{\perp_O}\ \mathrm{d}\delta & \text{if } r\sin\omega\delta + d_{\perp_0} > 0 \\ \int_{\delta_0}^{\delta_1} -r\sin\omega\delta + |d_{\perp_O}|\ \mathrm{d}\delta & \text{if } r\sin\omega\delta + d_{\perp_0} < 0 \end{cases} \tag{5.55}$$

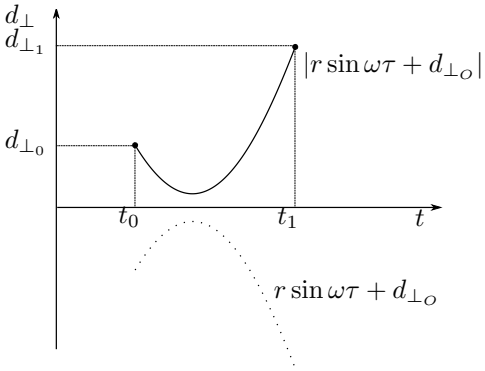
The cases where a crossing occurs require a combination of the two conditions shown in Equation 5.55, resulting in:

$$\int_{\delta_i}^{\delta_f} |r\sin\omega\delta + d_{\perp_O}|\ \mathrm{d}\delta = \int_{T_1} r\sin\omega\delta + d_{\perp_O}\ \mathrm{d}\delta + \int_{T_2} -r\sin\omega\delta + |d_{\perp_O}|\ \mathrm{d}\delta, \tag{5.56}$$

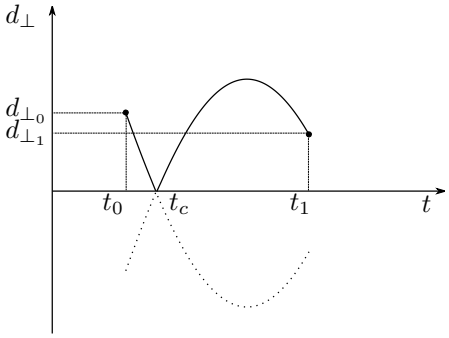
where T_1 is a set of all intervals in the positive domain defined by $r\sin\omega\delta + d_{\perp_0} > 0$, while T_2 represents all intervals in the negative domain, $r\sin\omega\delta + d_{\perp_0} < 0$, within the interval $[\delta_i, \delta_f]$. The path deviation cost can be determined for each manoeuvre sequence by adding the individual costs of each manoeuvre.



(a) A turn in the positive half plane



(b) A turn projected from the negative to the positive half plane



(c) Projections into positive half plane due to a turn crossing $d_\perp$

Figure 5.14: Representative path-deivation function for the additional horizontal manoeuvres

5.8 Search Optimisation Techniques

The implementation of an effective conflict resolution system requires real-time or near real-time execution. This section describes methods used to optimise the computational efficiency of the algorithm. These methods attempt to reduce the computational complexities that arise when dealing with a vast sampling space and therefore a large tree structure. Of the different functions required to develop a conflict-free, dynamically feasible path the conflict detection module and the LPM have been identified as the most computationally expensive. Of these, conflict detection can be considered the major bottleneck of the system [64]. Additionally the computation time required is proportional to the size and resolution of the sampling space [84, 85]. These factors have to be taken into account when attempting a real world application of the resolution algorithm. In an attempt to reduce the computational complexity of the algorithm, by minimising the number of function calls (calls to the modules described above), we have identified three techniques: Pruning, Rejection Sampling and a Dynamic Sampling Region. These search optimisation techniques are called when the algorithm has successfully determined a path to goal. This current path to goal's distance and cost is used by the search optimisation techniques to promote a fast convergence to an optimal solution.

5.8.1 Pruning

Pruning as defined in this section aims to reduce the computational complexity of the algorithm by reducing the number of vertices and edges in the tree structure. This reduction is achieved by removing the non-optimal vertices or pruning the tree. The main aim of the pruning functions is to avoid wasting time on low quality paths. This is achieved by reducing the size of the tree, thereby reducing the number of LPM and conflict detector calls. The implementation of the pruning function is important, because the adapted RRT* algorithm has a worst case LPM and conflict detector complexity of $O(n_S!)$, where n_S is the total number of samples. The pruning functions are only executed when a new path to the goal state has been determined. Once this path is found, the cost and distance to goal are used to successfully remove all vertices that cannot yield a more optimal path. The most basic form of pruning is cost-based node removal. Here the cost of the goal path is compared to the costs of each vertex in the tree. If a vertex possesses a higher cost, it is removed from the tree.

A more complex pruning implementation is the Projected Optimal path removal. This pruning method attempts to determine the optimal path, regardless of conflict or dynamic constraints, between each vertex in the tree and the goal state. The cost of these optimal paths is determined and combined with the current cost at each vertex. The resultant cost to goal is the lowest cost that can possibly be achieved from each vertex in the tree. These minimum costs are compared to the current goal path cost. If they exceed it they are pruned from the tree. This ensures that all the vertices left in the tree could at least theoretically promote a more optimal path and all vertices that could never yield an optimal path are removed. The projected optimal path removal function has been applied to the distance-based cost function only, using a straight line connection to goal method. The computation of the optimal paths to goal for the other cost functions is complex and no efficient method of goal path connection has been found.

5.8.2 Rejection Sampling

Rejection techniques were introduced by Kalos and Whitlock. They involve applying certain tests to the randomly sampled states, before allowing the continuation of the system [86]. Rejection sampling, as implemented in this system, is used in conjunction with the dynamic constraints of the aircraft to reduce unnecessary computation by removing non-flyable samples early. The aim of rejection sampling is to remove all samples that can be geometrically identified to not adhere to the dynamic constraints of the aircraft during the sampling process. This ensures that these samples can be removed early and no computation time is wasted on them. Rejection sampling should only remove samples that will definitely be rejected by the LPM. The implementation of rejection sampling requires the generation of a geometric shape that effectively describes the dynamic constraints of the aircraft. Therefore rejection sampling is applied only to the vertical sampling region, as theoretically any two states in the horizontal plane can be connected using the horizontal LPM, while the vertical constraints on the aircraft can effectively be represented geometrically.

Vertical Sampling Region

The sampling region, as defined in Section 5.5.1, has an elliptical form as seen in Figure 5.3. The rejection sampling region for the vertical domain is defined by the maximum climb and descent rates described by the

vertical LPM, discussed in Section 5.6.3. These rates are used to determine a rhombus-like acceptance zone that is applied to the elliptical sampling region and used to identify all dynamically realisable paths from the initial to the goal state as seen in Figure 5.15.

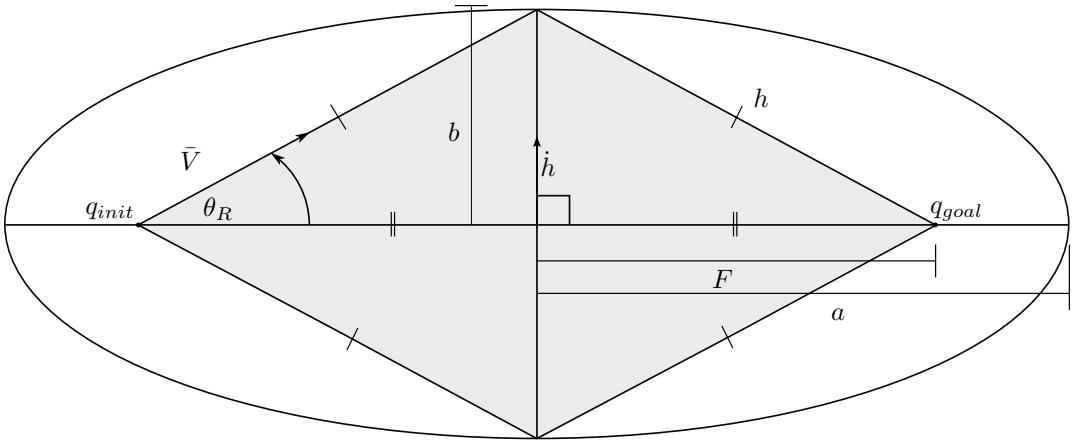


Figure 5.15: Graphical representation of the vertical acceptance zone in grey

The generation of the acceptance zone is determined by calculating the required acceptance angle θ_R from the maximum climb rate and the current speed of the aircraft. The acceptance angle is used in conjunction with the Euler distance between q_{init} and q_{goal} to produce the rhombus shaped acceptance zone in Figure 5.15. The generation of the acceptance zone is determined off-line to reduce computation time. The sampling process is adapted to sample states within the elliptical sampling region, and then test whether the sampled state should be rejected. If a state is accepted, in other words sampled within the acceptance zone (grey rhombus in Figure 5.15), then the algorithm continues. If not, the algorithm generates a new sample within the sample space. This process continues until a state is successfully sampled. Rejection sampling may have a low efficiency caused by rejecting many samples before one is accepted [86]. Therefore in order to implement an effective rejection sampler we must ensure that the acceptance and sample regions are constructed in such a fashion that more samples are accepted than rejected. We calculated that the worst case probability of rejection will always be lower than the probability of acceptance, given that $\theta_R \leq 31.75°$. Appendix C contains the mathematical calculations required to support this statement.

5.8.3 Dynamic Sampling Region – Sample Space Reduction

The implementation of a dynamic sampling region implies that the boundary of the elliptical sampling region, discussed in Section 5.5.1, can be adapted. This attribute can be utilised to improve the rate of optimal path discovery, thereby reducing the time required to reach an optimal path. This algorithm aims to improve the optimal path convergence rate, by reducing the search space based on the current optimal path's cost.

The dynamic sampling region makes use of a method of sample space reduction, reducing the size of the sampling region based on the distance of the current path to goal. Note from Section 5.5.1 that a favourable attribute of the elliptical sampling region is that we can define a constant maximum distance to the goal state via any sample within the space. This allows the implementation of a dynamic sampling region that reduces the sampling space as a function of the current path to goal distance.

Calculation of the New Sampling Region

The dynamically reduced sampling region's bounds are described as a function of the current path to goal distance P_d. Assume that an arbitrary path P has been determined to the goal state, with a path distance of P_d, as seen in Figure 5.16.

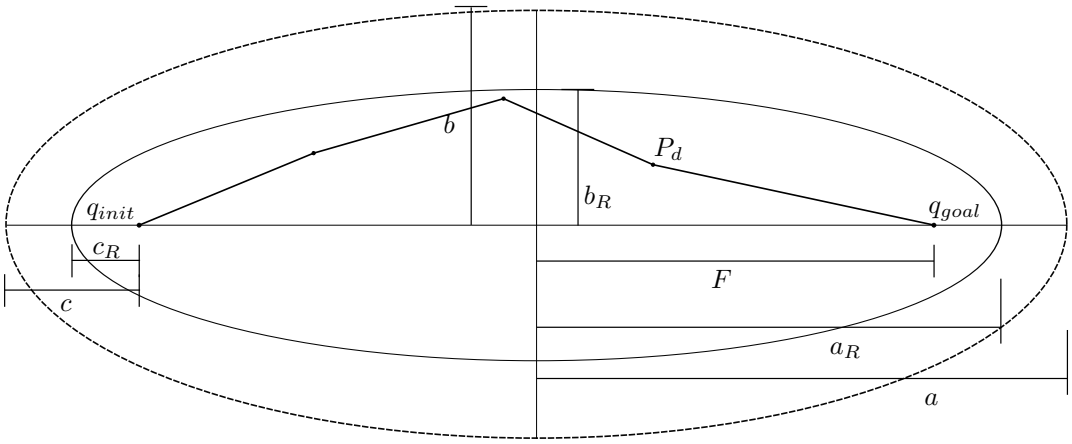


Figure 5.16: Elliptical Reduction applied based on the path to goal distance P_R

We can determine the new reduced semi major axis a_R using:

$$P_d = 2(c_R + F) \tag{5.57}$$

$$= 2a_R. \tag{5.58}$$

Once a_R is determined we make use of Equation 5.5 to determine the reduced semi-minor axis. This allows the redefinition of the sampling region after each new path to goal has been found.

5.8.4 Cross Dimensional Optimisation

Cross Dimensional Optimisation refers to the application of all the optimisation functions to both horizontal and vertical search trees once a path to goal has been determined, regardless of which tree contains the path. This implies that if a path to goal is found in the horizontal domain, both the horizontal and the vertical trees are optimised.

This functionality ensures that both the search trees strive to determine an optimal solution together. This is especially advantageous with regard to the implementation of a dynamic sample space (see Section 5.8.3) due to the nature of the different sampling regions. The horizontal sampling space is much larger than the vertical sampling region, due to the dynamic constraints of an aircraft (see Section 5.5.1). Therefore if a vertical path is found to the goal state, the horizontal sampling region is vastly reduced ensuring that no computation time is wasted on less optimal paths. This reduces the trade-off between the large horizontal search space and its optimal path convergence rate. By implementing the cross coupling of the search optimisation functions we enable the collective advance towards an optimal solution in both the horizontal and vertical domains.

5.9 Collaborative Resolution

The adapted RRT* as described in this section determines a conflict-free path to the goal state. The development of the algorithm assumes that it is only functional on the host aircraft, while all obstacle aircraft maintain their current trajectory or flight path. However, if the conflict resolution system is used by all aircraft in the environment, it vastly increases the complexity of the problem and potential complications can arise. If accurate information about the movement of all aircraft is available, and each aircraft is given a turn to resolve its conflict, the number of conflicts will reduce after each turn until, after all aircraft but the last one has had a turn, all conflicts will have been resolved. The resultant resolution, however, will not necessarily be globally optimal, since pairwise optimisation does not necessarily lead to global optimisation. However, this methods can be very time consuming and the application of additional logic to the algorithm could yield faster final path calculations. A potential complication that could arise is that all aircraft determine a resolution path at the same instant and therefore do not consider the new paths generated by the other aircraft when resolving the conflict. This problem is not handled or tested in this thesis. However we have provided some potential methods of dealing with this resolution problem.

1. A random starting time: the algorithm waits a random number of seconds before activating the resolution algorithm. This should cause the different aircraft to resolve the conflict at different times, therefore the system can receive the updated flight paths of the other aircraft before resolving the predicted conflict.
2. A designated resolution aircraft: assuming that communication between the aircraft is possible the different systems could designate a single resolution aircraft to resolve the conflict, while all others remain on their current flight paths.
3. Updating the flight path while resolving the conflict: while the resolution system determines new paths the system can update the current flight path of the aircraft. This constantly updated flight path can be broadcast to all obstacle aircraft and vice versa, thereby ensuring that all the resolution systems have the updated flight paths.

Finally a thorough test of the adapted RRT* algorithm is required to determine whether additions have to be made and what effect these additions have on the path calculations.

5.10 Summary

This chapter described the development of a conflict resolution system based on an adapted RRT* path planning algorithm and supporting functions, which include algorithms to implement local planning methods, cost functions for path optimisation, and search optimisation techniques. The adapted RRT* algorithm builds a tree structure rooted at a predefined initial state. During each iteration the algorithm samples a new point in space, within the sampling region, and attempts to connect it to the tree. If the connection is successful, meaning a flyable conflict-free path exists to the newly sampled state, the algorithm attempts to connect the sample to the goal state. After each successful goal state connection, numerous search optimisation

techniques are applied that adapt the sampling region, prune the tree and ensure that only the optimal path to the goal state is stored. When the algorithm's termination criteria are achieved, the best path to goal is returned. The algorithm's ability to derive this path to goal and ensure that it is optimal, flyable and conflict-free requires the application of different subsystems. The LPM module ensures that all paths derived are dynamically realisable, while a nominal conflict detection module is used to test for conflict. Search optimisation techniques are used to reduce the LPM and collision detector calls and promote faster convergence to an optimal solution. The development of each of these functions was described in this chapter as well as the assumptions and design choices required to derive the final conflict resolution system.

Chapter 6

Simulation and Results

The simulation of the algorithm aims to test the functionality of the adapted RRT* in static and dynamic environments. The algorithm is applied to four test scenarios; the generic two aircraft scenario, terrain only scenario, a two aircraft with terrain scenario and a multiple aircraft with terrain scenario. In each simulation the cost functions based on distance, energy and path deviation (described in Section 5.7) are applied. A statistical analysis of the paths generated by the algorithm is determined to test:

1. the ability to determine an initial conflict-free flyable path to the goal state,
2. the convergence rate of the algorithm towards an optimal solution,
3. the computational efficiency of the optimisation techniques,
4. and the integrity of the parallel generation of the search trees.

These tests aim to determine the applicability of the proposed algorithm as a path planner and ultimately as a potential conflict resolution system.

6.1 Simulation Setup

The algorithm was simulated using MATLAB 7.6.0 (R2008a) on a Windows 7 computer. The algorithm was applied using a Simulink model of the aircraft depicted in Figure 3.3. Each simulation was tested using a predefined initial and goal state, in the same horizontal plane. The simulations were set up to ensure that a conflict will occur between these states and that the goal state shall remain conflict-free. The applied system made use of a nominal conflict detection method described in Section 2.1.2. The simulations were applied assuming that the proposed conflict resolution system is only functional on the host aircraft ($\mathcal{H}$), while all obstacle aircraft ($\mathcal{O}$) remain on their predicted trajectories. An estimation of the optimal solution is determined through application of a 10 000 iteration (I_S) simulation. During one iteration the algorithm

samples a point each in the horizontal and the vertical domain, therefore one iteration corresponds to two sampled points. The estimated optimal solution is assumed to be optimal due to the asymptotic optimality of the adapted RRT*; the validity of this assumption is discussed in Section 6.2.1. A statistical representation of the expected behaviour of the algorithm was determined through the use of 1000 simulations.

6.1.1 Generic Two Aircraft

The generic two aircraft scenario is a dynamic conflict example, generally used as a benchmark for conflict avoidance tests. The scenario, illustrated in Figure 6.1, occurs between two aircraft with perpendicular intersecting trajectories or flight plans. A conflict is detected T seconds in the future, where T is the look ahead time of the system. This scenario was selected to test the ability of the conflict resolution system in a relatively simple dynamic environment.

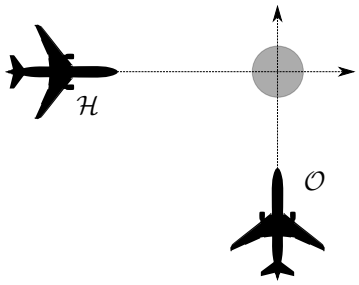
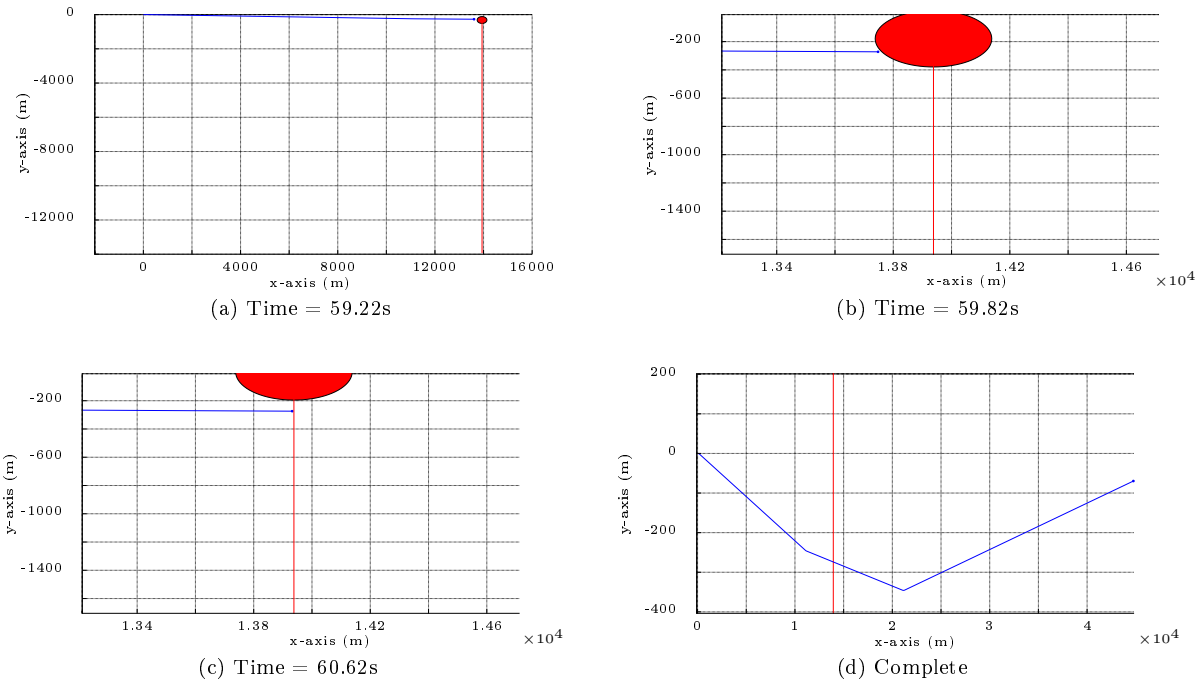


Figure 6.1: Generic two aircraft conflict scenario

6.1.1.1 Paths Generated

The adapted RRT* was applied to the generic two aircraft scenario and the paths generated when applying each of the cost functions are depicted by Figures 6.2, 6.3 and 6.4. These figures show the progression of the developed path at different time instances. The system is activated when a conflict is predicted 60 seconds in advance and a resolution path is determined. The obstacle aircraft is represented by the red sphere, while the paths flown by the host and obstacle aircraft are shown by the blue and red lines respectively. The spherical obstacle aircraft is depicted as a flattened ellipse in most of the figures. This is because the dimensions of the axes are not equal and the x-axis is generally much longer than both the y- and z-axis.

Figure 6.2 shows the development of a horizontal resolution path, each of the sub-figures show how the path avoids the oncoming obstacle aircraft (red sphere). The host aircraft avoids the obstacle by a very small distance, showing that the distance-based cost function determined a near optimal solution. In Figure 6.2d it is possible to see the exact resolution path determined by the algorithm.



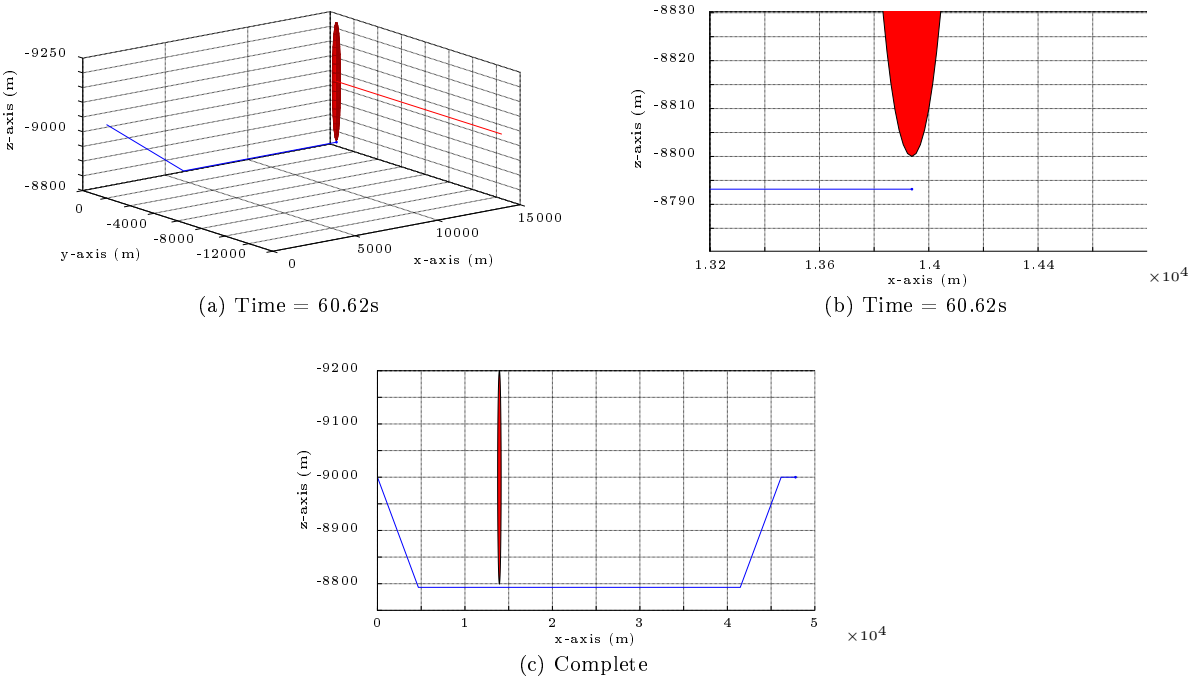
(a) Time = 59.22s

(b) Time = 59.82s

(c) Time = 60.62s

(d) Complete

Figure 6.2: The paths generated, at different time instances, for the generic two aircraft scenario when applying the distance-based cost function for 100 algorithm iterations

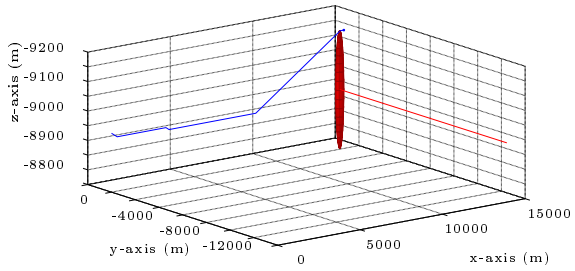


(a) Time = 60.62s

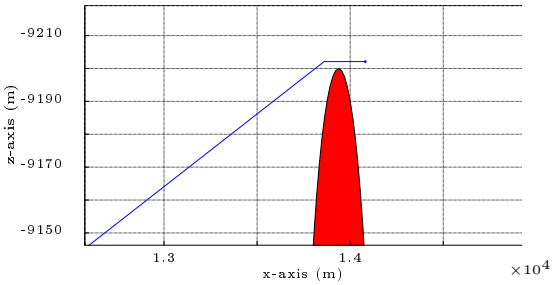
(b) Time = 60.62s

(c) Complete

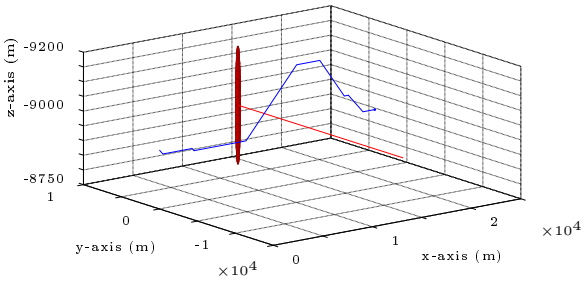
Figure 6.3: The paths generated, at different time instances, for the generic two aircraft scenario when applying the energy-based cost function at 100 iterations



(a) Time = 61.22s



(b) Time = 61.22s



(c) Complete

Figure 6.4: The paths generated, at different time instances, for the generic two aircraft scenario when applying the path deviation-based cost function at 100 iterations

The application of the energy-based cost function to the generic two aircraft scenario is depicted in Figure 6.3. This resulted in a vertical resolution path as expected, because the extra energy required for a climbing manoeuvre is equal to the energy saved during an equivalent descending manoeuvre. This implies that less energy is generally required to execute vertical resolution (see Section 6.2.1 for details). The path deviation-based cost function depicted in Figure 6.4 shows how the algorithm attempts to deviate from the intended trajectory as late as possible and return as soon as it can. From the simulation results for the three cost functions, it is clear that the distance- and energy-based cost functions determine paths that are easier for a pilot to follow, as these paths consist of only a few manoeuvres. The resolution path generated by the path deviation-based cost function consists of multiple manoeuvres and post-processing is required to simplify this path.

6.1.2 Terrain Only

The terrain only scenario tests the algorithm's ability to derive a conflict-free path in a static environment. The simulation is set up so that if the host aircraft remains on its current trajectory a collision with a mountainous terrain will occur (illustrated in Figure 6.5). For this test the mountainous terrain is set up so that vertical resolution is encouraged. This is accomplished by ensuring that the height of the mountain, relative to the aircraft, is less than its relative width. This ensures that vertical resolution is preferred, at least for the distance-based cost function.

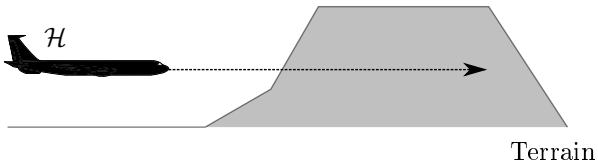
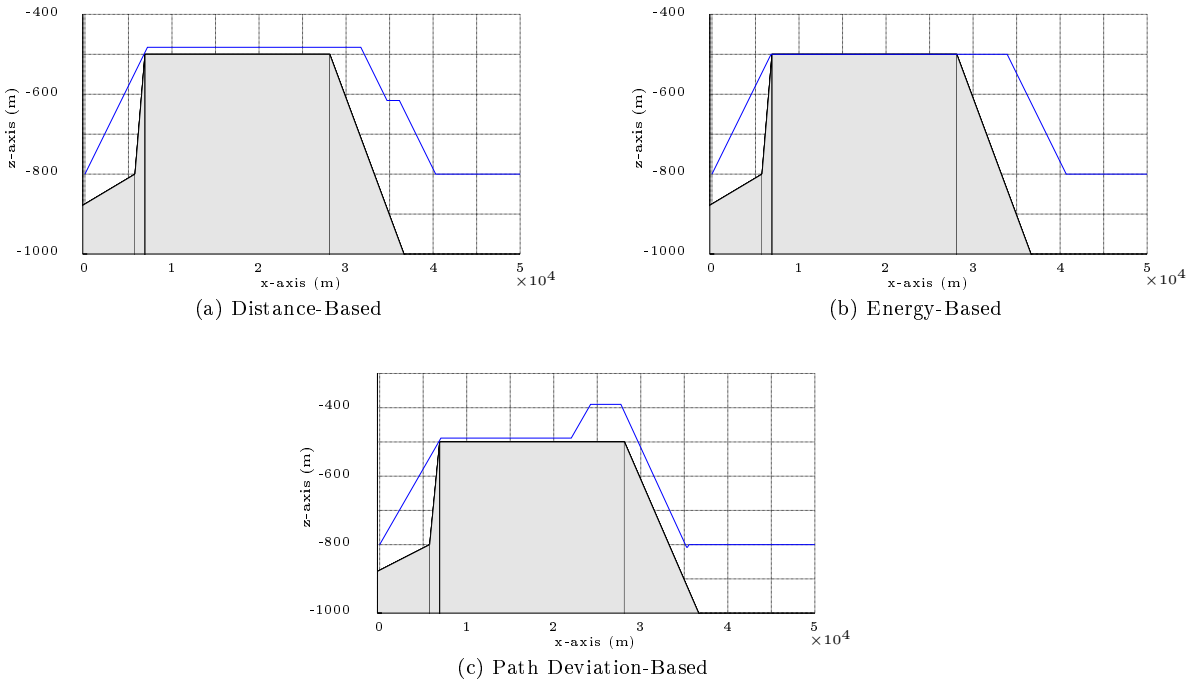


Figure 6.5: Terrain only conflict scenario

6.1.2.1 Paths Generated



(a) Distance-Based

(b) Energy-Based

(c) Path Deviation-Based

Figure 6.6: The complete paths generated for the terrain only scenario at 100 iterations

The paths generated during a 100 iteration simulation of the terrain only scenario are depicted in Figure 6.6. Each of the resolution paths determine a solution that avoids the conflict region (grey area). The paths generated by the distance- and energy-based cost functions seem to make use of less manoeuvres, while the path deviation-based cost function exhibits a unexpected climb above the mountain as seen in Figure 6.6c. This climb is required so that the system can descend to the original path earlier. This is because it is not possible for the path to connect to the goal state immediately from the sample before the climb. However, the sample at the end of the climb and straight segments can connect to the goal state, thereby resulting in the least deviation from the nominal path.

6.1.3 Two Aircraft with Terrain

The single aircraft with terrain scenario aims to test the performance of the algorithm in an environment that contains both static and dynamic obstacles. This scenario is set up by making use of the mountainous terrain in the terrain only scenario, with the addition of an obstacle aircraft flying above the mountain as

shown in Figure 6.7. The dimensions of the mountainous terrain, as described above, ensure that the optimal resolution path is to fly over the mountain, but in this scenario such a path will result in a conflict with an obstacle aircraft. This forces the algorithm to derive a horizontal solution, as it is the only safe path that avoids both the terrain and aircraft.

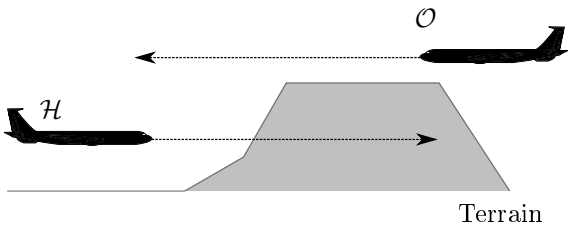
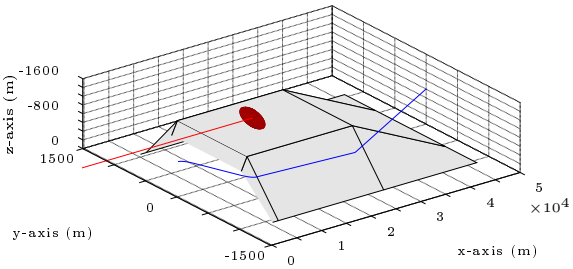
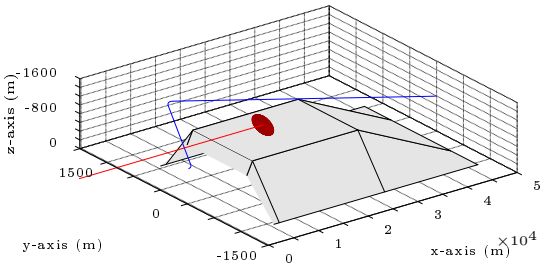


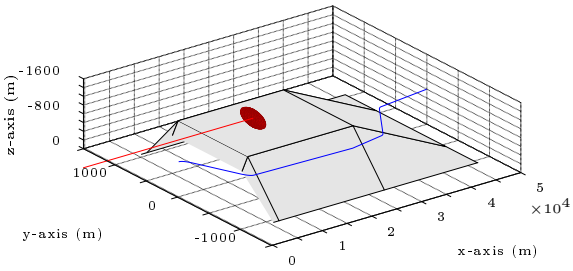
Figure 6.7: Two aircraft with terrain conflict scenario



(a) Distance-Based



(b) Energy-Based



(c) Path Deviation-Based

Figure 6.8: The complete paths generated for the two aircraft with terrain scenario at 100 iterations

6.1.3.1 Paths Generated

The paths generated for the two aircraft with terrain scenario all resulted in horizontal manoeuvres, due to the obstacle aircraft flying above the mountain. It is clear from Figure 6.6 that the optimal path without the obstacle aircraft was a vertical manoeuvre over the terrain. However, as seen in Figure 6.8, no vertical resolution was possible due to the presence of the obstacle aircraft and all the simulations determined a horizontal path regardless of the applied cost function. Figure 6.8 shows that all the paths generated by the algorithm are simple to follow, consisting of only a few manoeuvres. The characteristics of the different cost functions are exhibited in the paths generated. The distance-based cost function, shown in Figure 6.8a, attempted a straight connection to the goal state after passing the mountain, thereby minimising the

distance travelled. The energy-based cost function made use of as little turning manoeuvres as possible to reduce the energy consumed during a turn, as seen in Figure 6.8b. Finally the path deviation-based cost function, depicted in Figure 6.8c, determined a path that hugs the mountainside in order to reduce the distance deviated from the nominal path.

6.1.4 Multiple Aircraft with Terrain

The multiple aircraft with terrain scenario makes use of the same set up that was used in the two aircraft with terrain scenario, with the addition of three obstacle aircraft (as seen in Figure 6.9). This scenario tests the performance of the algorithm in a cluttered dynamic environment containing static terrain. This scenario is used as a representation of a cluttered, dynamic airport environment.

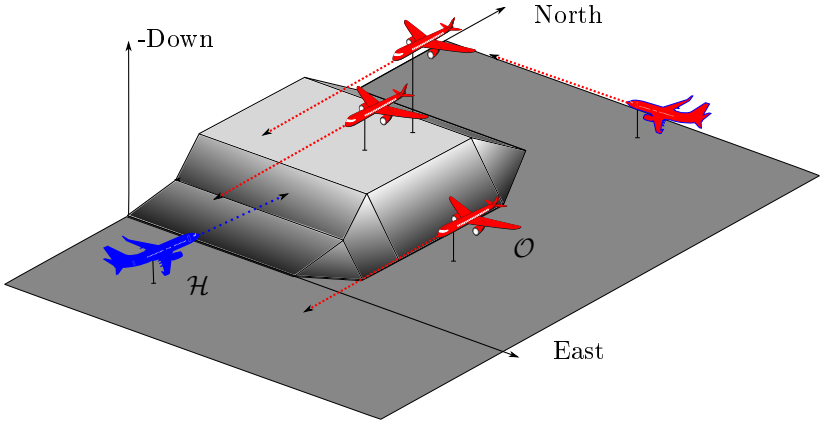


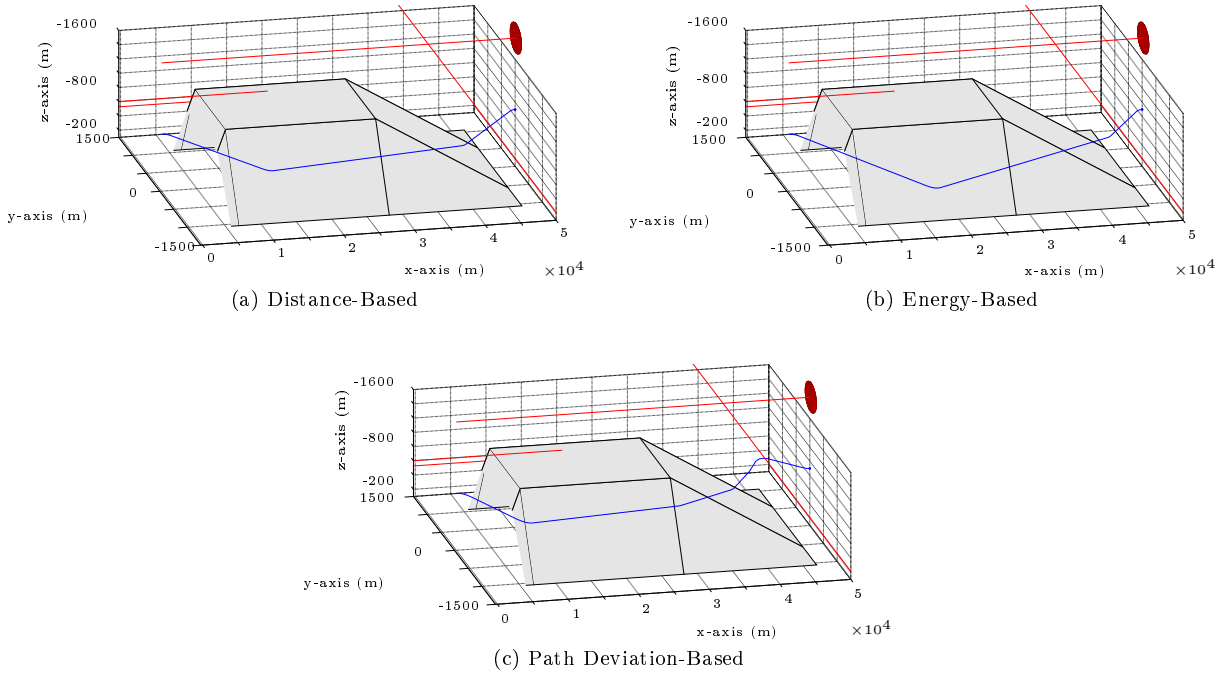
Figure 6.9: Multiple aircraft with terrain conflict scenario. The obstacle aircraft are depicted in red, while the host aircraft is in blue.

6.1.4.1 Paths Generated

The paths generated during the multiple aircraft with terrain scenario are similar to those determined during the two aircraft with terrain simulations. This is due to the similarity between the scenarios. However as seen in Figure 6.10c the path determine when applying the path deviation-based cost function required additional manoeuvres to avoid the obstacle aircraft North of the mountain.

6.2 Results

This section discusses the statistical interpretation of the results obtained from the simulations. The algorithm derives a search tree for each scenario; representations of the trees calculated are given in Figure 6.11. The figure depicts the search tree calculated in the horizontal (Figure 6.11a) and vertical (Figure 6.11b) domains for the terrain only scenario. Here the algorithms attempts to avoid a static mountainous terrain. The search trees are comprised of red edges and blue vertices. The calculated paths to the goal state are shown in blue, while all vertices removed from the trees by the numerous search optimisation techniques are in green.



(a) Distance-Based

(b) Energy-Based

(c) Path Deviation-Based

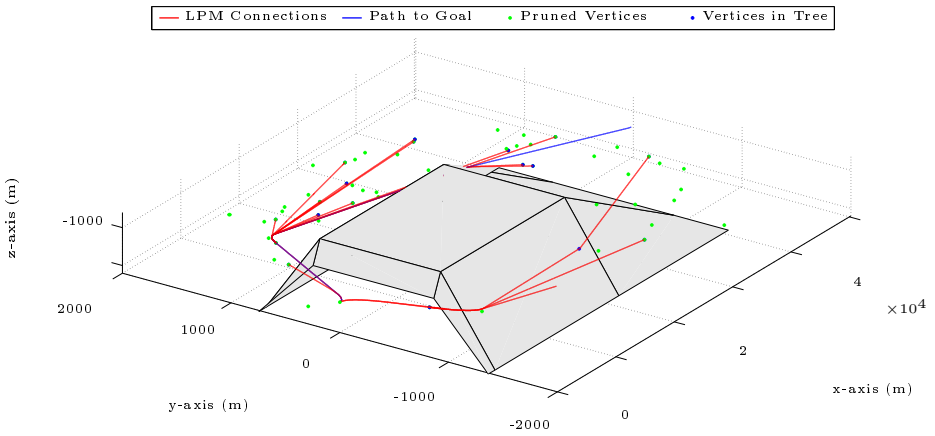
Figure 6.10: The complete paths generated for the mulitple aircraft with terrain scenario at 100 iterations

The search trees obtained above are created by randomly sampling the search space. Figure 6.12 is a graphical representation of the samples generated in the vertical domain. The figure depicts the random samples (blue), as well as the reduction in the dynamic sampling space size (red). Figure 6.12 shows that the samples generated by the algorithm are concentrated within the optimal sampling region and that this optimal sampling region is obtained within a few samples.

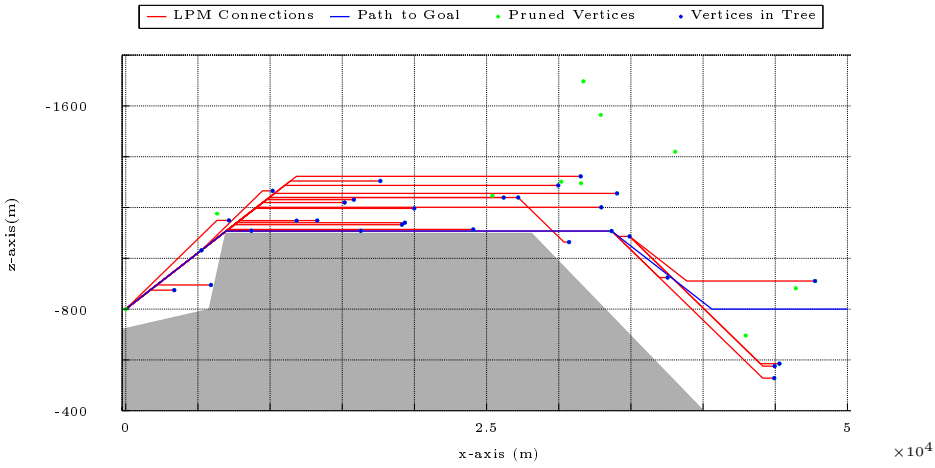
The sections that follow analyse the ability of the algorithm to determine an initial path to the goal state, while also looking into the convergence rate of the algorithm towards an optimal solution. Additionally an analysis of the different search optimisation techniques applied to the system as well as the influence of sequential and parallel development of the search trees are described. The algorithm's ability to determine a path in real or near real-time was not analysed. This is because the algorithm could not be tested on the expected implementation platform. A timing analysis in MATLAB on a Windows computer is impractical since MATLAB is a high-level interpreted language, which is very inefficient with regard to computation time. An effective timing analysis requires the application of the proposed conflict resolution system in conjunction with the external conflict detection module on a representative aircraft computer system using the appropriate coding language. Unfortunately this was not possible and therefore no timing analyses were conducted. However the relative reduction in function calls were analysed, where applicable.

6.2.1 Simulated Optimal Cost

The asymptotic optimality of the adapted RRT* ensures that as the number of samples increases the paths generated will converge towards an optimal path. However, this does not ensure that a simulation with



(a) Horizontal Search Tree



(b) Vertical Search Tree

Figure 6.11: Graphical representation of the search tree's calculated to traverse the terrain only scenario

$I_S = 10000$ is an accurate representation of the optimal solution, where I_S represents the number of iterations. This section evaluates the accuracy of this representation, by calculating the optimal path cost mathematically and comparing it to the simulated optimal cost. The optimal costs are calculated for the generic two aircraft scenario, as a geometric representation of the optimal paths is simple to derive for this scenario. A discussion on how these geometrically optimal paths are calculated is given below.

The distance-based cost function aims to minimise the path distance travelled between the initial and goal state. Therefore, the expected optimal result derived by the algorithm is a horizontal path that makes use of the smallest angle required to avoid the predicted conflict and return to the goal state. This geometrically optimal resolution path is depicted in Figure 6.13a.

The energy-based cost function strives towards a path that requires the minimal amount of energy to execute. The formulation of the energy-based cost function derives the standard straight and level energy required and adds all the additional energy used when executing a turning or climbing manoeuvre. A descent manoeuvre, however, makes use of less energy than straight and level flight. The amount of energy saved during a

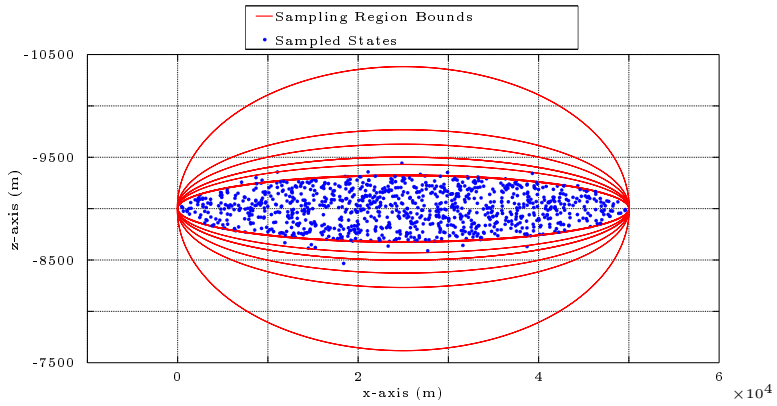
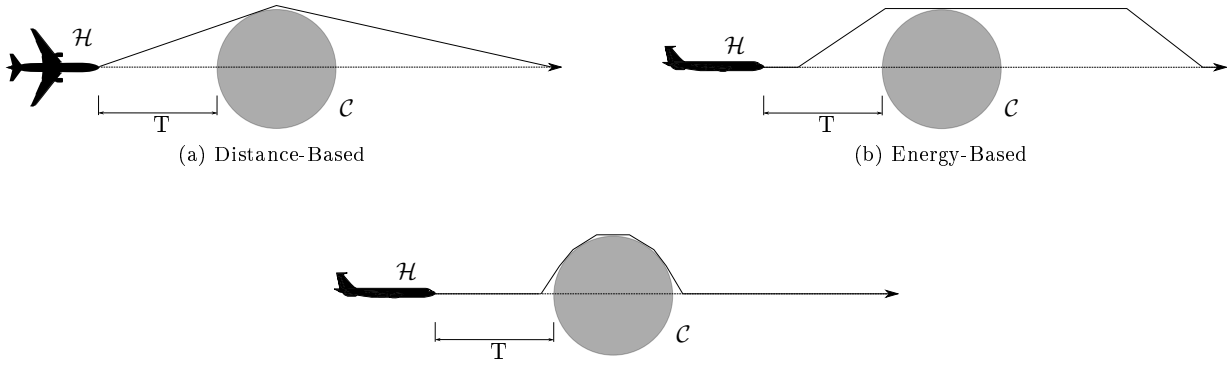


Figure 6.12: Samples generated by adapted RRT*, with a dynamic sampling region



(a) Distance-Based

(b) Energy-Based

(c) Path Deviation-Based

Figure 6.13: Geometric representations of the optimal path for each of the cost functions, applied to the generic two aircraft scenario. The scale has been adapted for representation purposes.

descent is equal to the additional energy required to ascend at the same rate. This implies that ascending and descending manoeuvres can collectively be viewed as a straight and level flight manoeuvre, if the climb and descent rates are equivalent. The optimal energy-based path for the generic two aircraft scenario is therefore vertical resolution that results in the shortest path distance or time, as shown in Figure 6.13b.

The path deviation-based cost function aims to remain on the current path as long as possible. The optimal path can be defined as the solution with the minimal time integral of the path deviation, or with the smallest deviation from the original path, for the shortest time. The expected optimal solution is therefore one that makes use of the steepest climb and descent rates as late as possible, resulting in the resolution path depicted in Figure 6.13c.

The simulated costs, at $I_S = 10000$, were compared to the geometrically calculated optimal costs. The deviations between them are depicted in Table 6.1. The distance and energy-based simulations exhibited near-optimal solutions with very slight deviations from the geometrically optimal costs. However, a maximum deviation of 10.8204 % was recorded when applying the path deviation-based cost function. This deviation is relatively large and implies that a larger number of samples is required to obtain an optimal solution for the path deviation-based cost function. When looking at the distance and energy-based cost functions, the selected number of samples resulted in a path that can be considered an accurate representation of the

Table 6.1: Deviation of the simulated optimal solution at $I_S = 10000$ from the geometrically calculated optimal cost, for the generic two aricraft scenario.

Cost Function	Simulated cost's deviation from geometrically calculated path cost
Distance	0.0038 %
Energy	0.000002 %
Path Deviation	10.82 %

optimal solution. Due to the negligibly small deviations for both the distance and energy-based cost functions, $I_S = 10000$ was selected as the optimal path representation for all the cost functions, to ensure uniformity between the tests. The slow convergence rate of the path deviation-based cost function is discussed in Section 6.2.3. The simulated cost for 10000 iterations will be used as if it were the optimal cost, and will be referred to as such for the remainder of this chapter.

6.2.2 First Path Calculations

The primary requirement for any conflict resolution system is the fast calculation of a differentially realisable, conflict-free path to the goal state of the system. The aim of the adapted RRT* algorithm is to determine this initial path to goal within as few samples as possible. The number of samples required to determine the initial path to goal for each scenario over the 1000 simulations is shown in Table 6.2. The table shows the maximum and mean ($\bar{n}$) number of samples required to determine the initial path to goal. Additionally the standard deviation (σ_n) of the number of samples required to determine the initial path to goal is also shown.

The mean number of samples required to generate an initial path to the goal state remains constant for each scenario. This shows that the number of samples required to find the first path to the goal state is independent of the cost function used to optimise the path. The selection of a cost function will therefore not impede the algorithm's ability to obtain an initial safe path. This statement is bolstered by the similar standard deviations within each scenario.

The type of cost function may not affect the initial path to goal calculations, but there does exist a relationship between the number of samples required to find the first path and the specific conflict scenario. Table 6.2 shows that as the complexity of the scenario increases, the maximum number of samples required increases. However, the increase in the mean number of samples remains very small, with a difference of only 0.813 between the maximum and minimum mean number of samples across all simulations. Although the maximum number of samples required to find the first path does increase as the complexity of the conflict scenario increases, the maximum number of samples required remains below 7 for 99.7% of the simulation runs (3σ band), for all conflict scenarios considered.

The generation of an initial path within a few samples for all test scenarios is very important to ensure that a conflict resolution system is viable. The results depicted in Table 6.2 show that the adapted RRT* adheres to this requirement, deriving initial paths to the goal state quickly and efficiently.

Table 6.2: Statistical results depicting the samples required to calculate the first path to goal, determined from 1000 simulations at 100 iterations for each scenario.

Cost Function	Number of Samples (n)		
Generic Two Aircraft	Max	$\bar{n}$	σ_n
Distance	3	1.072	0.321
Energy	3	1.059	0.282
Path Deviation	6	1.081	0.356
Terrain Only			
Distance	8	1.61	1.167
Energy	8	1.597	1.142
Path Deviation	9	1.658	1.192
Two Aircraft, with Terrain			
Distance	17	1.848	1.658
Energy	11	1.872	1.524
Path Deviation	13	1.822	1.54
Multiple Aircraft, with Terrain			
Distance	9	1.66	1.828
Energy	11	1.673	1.256
Path Deviation	9	1.644	1.242

6.2.3 Convergence to an Optimal Path

The secondary aim of the adapted RRT* algorithm presented in this thesis is to utilise the remaining execution time, after an initial path to the goal state has been determined, to strive towards an optimal solution. This section analyses the algorithm's ability to obtain optimal solutions by looking at the average optimal costs achieved as well as cost function's convergence rate towards an optimal solution.

The relationship between the mean path costs, at $I_S = 100$, and the optimal solution c^* is depicted in Table 6.3. This data is derived from histograms of the optimal path costs achieved over a large number of different simulation runs, for all three cost functions, and in all four conflict scenarios. The histograms for the generic two aircraft scenario are shown in Figure 6.14. The histograms for the other three conflict scenarios are similar, and can be found in Appendix D. The percentage difference between the mean costs ($\bar{c}$) calculated during each of the 1000 simulations and the optimal cost is shown in the table. The percentage standard deviation $\sigma_{\%}$ from $\bar{c}$ is also supplied. Figure 6.15 depicts the percentage decrease from the optimal path cost as the number of samples increases. All test scenarios resulted in interchangeable graphs exhibiting a similar general shape or trend. The generic two aircraft scenario's results shown in Figure 6.15, were selected as a general representation of the cost trends for all the simulations. The median cost at each sample is depicted in blue, while the 68th, 95th and 99.7th percentiles are also given. The graph depicts an exponential-like decay in cost tending towards the optimal cost c^*, at 0%. The graphs depicted in the figure are substantiated by Table 6.3 as the graphs decay towards the expected average costs for each simulation.

Table 6.3 shows that the distance and energy-based cost functions exhibit similar average final cost charac-

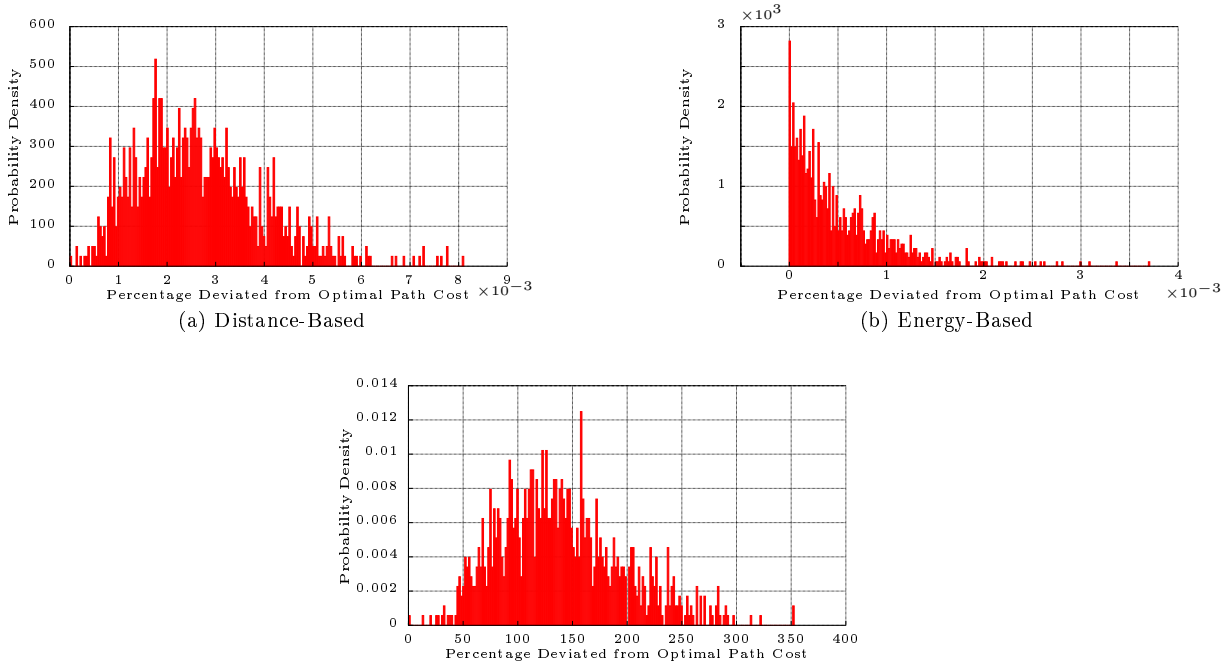


Figure 6.14: Histograms of the optimal path cost pdf, for the generic two aircraft scenario at $I_S = 100$, normalised as a persentage from the optimal solution determined at $I_S = 10000$.

teristics, converging towards near-optimal solutions within 100 iterations. The deviation for both these cost functions at 100 iterations is negligibly small for all practical implementations. The convergence rate of these two cost functions can be considered fast enough that a practically optimal solution is reached within 100 iterations. This is substantiated by Figures 6.15a and 6.15b.

The path deviation-based cost function exhibits a slower convergence rate, shown in Table 6.3. When comparing Figures 6.15c and 6.15d it is clear that the algorithm still converges towards an optimal solution, but at a reduced rate. The slow convergence rate could be attributed to the search optimisation techniques, especially the application of the dynamic sampling space. The distance and energy-based cost functions are directly coupled with time (or distance) (see Equations 5.19 and 5.44). Additionally the dynamic sampling region is reduced based on the path distance (i.e. time), therefore it has is a strong connection to the distance and energy-based cost functions. The application of the dynamic sampling region therefore improves the convergence rates of the distance and energy-based cost functions, while having little influence on the time-deviation cost function. The application of specialised cost function based search optimisation techniques, should improve the convergence rate of the algorithm. The various search optimisation techniques are analysed in Section 6.2.4.

The cost convergence graphs, seen in Figure 6.15, allow the calculation of the algorithmic decay rate to an optimal solution. This enables the estimation of the maximum number of samples required to ensure near-optimal path calculations, for each specific cost function.

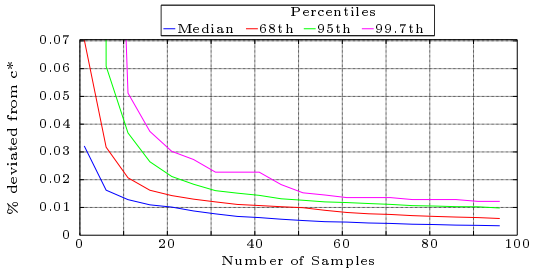
Table 6.3: Statistical results depicting optimal path characteristics, determined from 1000 simulations at 100 iterations for each scenario.

Cost Function	**Average Cost ($\bar{c}$)**	
Generic Two Aircraft	% of c^*	$\sigma_{\%}$
Distance	100.003	0.00128
Energy	100.0005	0.00052
Path Deviation	270.432	22.751
Terrain only		
Distance	100.0029	0.003
Energy	100.003	0.0032
Path Deviation	124.41	9.531
Two Aircraft, with Terrain		
Distance	100.07	0.045
Energy	100.223	0.147
Path Deviation	184.951	19.779
Multiple Aircraft, with Terrain		
Distance	100.254	37.257
Energy	102.049	12.829
Path Deviation	428.573	29.151

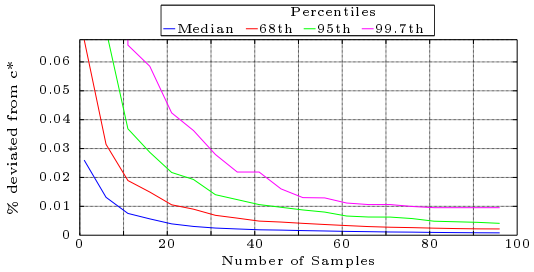
6.2.4 Search Optimisation Techniques

The search optimisation techniques introduced in Section 5.8 aim to reduce the computation time of the algorithm, by reducing the number of conflict detector and LPM calls required to obtain a solution. The major bottleneck for all path planning systems can be considered to be the conflict detection module [64]. Due to this bottleneck we make use of the number of conflict detection module calls as a measure of the computational efficiency or computational complexity of the algorithm. This implies that in order for the search optimisation function to effectively reduce the algorithmic execution time it must minimise the number of conflict detection module calls. To test this, the algorithm has been applied to each scenario with and without the search optimisation techniques. In order to make a fair comparison of their computational efficiencies, the path planning algorithms with and without search optimisation are both executed until they reach the same fixed predetermined path cost. This termination cost was determined from the average cost percentages of the optimal solution at 10 iterations in Table D.1, Appendix D for each simulation, thus the two systems will both yield an equally optimal path. The termination cost at 10 iterations was selected, because some simulations without the search optimisation techniques required up to 700% more samples as seen in Table 6.4, resulting in very long execution times. A comparison between the number of conflict detection module calls will therefore provide a measure of the effectiveness of the search optimisation techniques applied. The results determined from the aforementioned tests are depicted in Table 6.4.

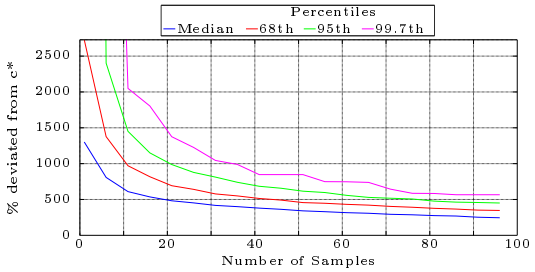
The table depicts the percentage difference for each simulation and cost function between the system applied with and without the optimisation techniques. It can be noted that each simulation without search optimisation required more samples to reach the termination path cost. Additionally each of these simulations



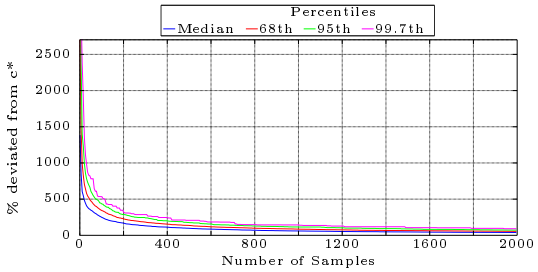
(a) Distance-Based ($I_S = 100$)



(b) Energy-Based ($I_S = 100$)



(c) Path Deviation-Based ($I_S = 100$)



(d) Path Deviation-Based ($I_S = 2000$)

Figure 6.15: Statistical cost convergence of the median cost for the generic two aircraft scenario over 1000 simulations

required more LPM function calls with a minimum average increase in LPM calls of approximately 100% across all simulations. The conflict detector calls used to represent the time required for each simulation varied with each scenario and cost function. A maximum average increase in conflict detection calls, approximately 430%, was recorded during the generic two aircraft scenario, when applying a distance-based cost function, while all simulations required more conflict detector function calls, although these did vary dramatically. These results give a clear indication that the application of the search optimisation techniques reduced the computational complexity for each simulation. The tests showed that the application of these techniques, regardless of the cost function or scenario, improve the efficiency of the algorithm, promoting the faster generation of more optimal paths. The effect of the search optimisation techniques is more pronounced in simpler scenarios, where a conflict is predicted for a single instant as seen in the generic two aircraft simulation. It is clear that the computational benefit provided by the search optimisation techniques are scenario and cost function dependent and that the complexity of the scenario affects the ability of these techniques. However, the cause of these differences is not clear.

An analysis of the standard deviations for each scenario and cost functions showed that when the search optimisation techniques are applied, the variance of the total execution time is smaller. This ensures that the expected computation time when applying the search optimisation techniques can be estimated more accurately. Overall the application of the search optimisation techniques depicted a reduction in all function calls. This ensures faster execution times as well as increased confidence in the expected completion time.

Table 6.4: Statistical results depicting the effect of the search optimisation techniques on the final path calculations, determined from 1000 simulations. Each simulation was terminated when the desired path cost was reached. The table shows the percentage difference in the required samples and function calls when no search optimisation techniques are applied.

Cost Function		% **Difference in Function Calls**	
Generic Two Aircraft	% difference in I_S	LPM	Conflict Detector
Distance	733.471	14785.445	432.192
Energy	119.422	425.959	29.612
Path Deviation	136.835	734.477	50.016
Terrain Only			
Distance	55.865	220.689	44.916
Energy	58.206	173.905	46.372
Path Deviation	65.542	316.797	60.823
Two Aircraft, with Terrain			
Distance	20.507	97.668	20.289
Energy	36.808	127.541	37.502
Path Deviation	74.057	368.214	86.213
Multiple Aircraft, with Terrain			
Distance	20.507	97.667	20.289
Energy	36.808	127.541	37.502
Path Deviation	368.214	96.614	86.213

6.2.5 Sequential versus Parallel Path Calculation

This section verifies the validity of the assumption that the parallel development of the search trees, described in Section 5.3.1.1, is more efficient than sequential execution. Each scenario is tested by sequentially determining a vertical and horizontal path and selecting the optimal solution. This solution is compared to the parallel implementation of the adapted RRT*. The results of these simulations are depicted in Table 6.5. The table shows the relationship between the parallel and sequential execution's average final costs and function calls. These are depicted as the percentage difference between the average sequential and parallel execution results.

All three simulations show that both parallel and sequential execution strive towards the same optimal solution, with a maximum deviation in cost of less than 1.051%. Sequential execution generally required more LPM function calls, with a maximum increase of 96.7%. The parallel execution of the search trees promoted fewer function calls with respect to both the LPM and the conflict detection module. As the complexity of the scenarios increases the effect of parallel and sequential execution on the function calls and samples required reduces, as seen in the multiple aircraft with terrain. Generally sequential resolution required more conflict detection module calls, barring the distance-based generic two aircraft simulation. In all other simulations the parallel development of the trees resulted in a reduction in the number conflict detection module calls. This scenario, however, resulted in $\approx 13\%$ fewer conflict detection module calls when sequential resolution was applied. A possible reason for the reduced conflict detector calls when executing the algorithm sequentially could be caused by cross dimensional optimisation (seen in Section 5.8.4) applied during the parallel implementation. The geometry of the scenario and cost function strongly promotes a

Table 6.5: Statistical results highlighting the difference between sequential and parallel development of the search trees, determined from 1000 simulations at 100 iterations for each scenario.

Cost Function		% **Difference in Function Calls**	
Generic Two Aircraft	% Difference in cost	LPM	Conflict Detector
Distance	0.00013	15.608	-12.938
Energy	0.00001	41.72	10.162
Path Deviation	1.05	35.3547	35.355
Terrain Only			
Distance	-0.00006	39.444	7.216
Energy	-0.0002	53.98	8.433
Path Deviation	-0.129	83.084	15.544
Two Aircraft, with Terrain			
Distance	-0.004	-0.752	1.156
Energy	-0.0077	0.194	-0.151
Path Deviation	0.275	0.69	0.956
Multiple Aircraft, with Terrain			
Distance	0.0001	59.22	5.849
Energy	0.0001	62.33	2.85
Path Deviation	0.592	96.614	11.205

horizontal path; this could potentially result in such a large reduction in the vertical sample space that all sampled states will result in a conflict, thereby increasing the conflict detection module calls dramatically. Further tests are required to determine whether this is indeed the case. If it is, it could be remedied by only connecting each newly sampled state to vertices in the tree within a predefined spherical region around the sample. This ensures that the size of the tree and a high probability of conflict to that sample do not result in a bottleneck for the system. This connection method was implemented by Karaman and Frazzoli in the original RRT* [62].

All the aircraft with terrain simulations resulted in similar costs, and function calls. However no vertical resolution path could be calculated within 100 iterations. The application of sequential resolution resulted in the algorithm spending unnecessary time attempting to derive a vertical solution, while a feasible horizontal path could easily be calculated. The computation times required to obtain the final path for both sequential and parallel implementations can be considered equal, but the order in which the sequential resolution system searches the C-space can affect the algorithm's ability to derive an initial path quickly. Both parallel and sequential executions resulted in equivalent optimal path calculations.

The results depicted in Table 6.5 show that parallel execution can generally be considered a more computationally efficient implementation of the adapted RRT* algorithm. However, scenarios do arise where sequential resolution can be considered equally or even more efficient, as seen with the distance-based, generic two aircraft scenario. Finally, the main advantage of the parallel development of the search trees is the fast derivation of an initial path to goal, regardless of the geometry of the environment. This can not be guaranteed when executing the searches sequentially, as the search order will affect the initial path calculation.

6.2.6 Paths Generated

The paths generated by the adapted RRT* are generally comprised of a few vertices or way-points. This is due to the convergence of the paths towards an optimal solution. The selected cost functions promote simple optimal paths consisting of two or three manoeuvres, ensuring that the algorithm strives to determine simple paths. This makes it easy for the pilot to follow the proposed resolution paths. Examples of the paths generated by the algorithm are depicted in Figures 6.2, 6.3, 6.4, 6.6, 6.8 and 6.10. The simplicity of the paths obtained when applying the distance and energy-based cost functions once again show that these methods strive towards an optimal solution at a higher rate. The path deviation-based cost function generally determines more complicated paths consisting of multiple manoeuvres, this can be reduced by applying a post processing function that linearises the proposed paths. Overall the paths derived by the algorithm are simple to follow and resolve the predicted conflict. The paths developed do not exhibit unexpected manoeuvres and seem similar to resolution paths that we expect a human pilot to derive when faced with equivalent conflict scenarios.

6.3 Summary

The adapted RRT* was applied to multiple simulations, consisting of 4 different conflict scenarios of increasing complexity. In all the simulations a safe path to the goal state was determined, showing that the algorithm can safely determine a feasible resolution path in different situations. These initial paths to the goal state were determined quickly requiring 7 samples to reach the goal state during 99.7% of the simulations (3σ band). The fast calculation of an initial path to the goal state is very important for the application of the system as a conflict resolution module. The algorithms convergence rate was also analysed to ensure that the system converges towards an optimal solution. The results obtained were from these tests showed that the algorithm exhibits and exponential-like decay towards an optimal solution as the number of samples increases. A high convergence rate was observed when applying the distance- and energy-based cost functions, with the algorithm converging to a practically optimal solution well before a 100 samples. The application of the path deviation-based cost function exhibited a slower rate of convergence and further study is required to determine the exact cause of this.

Finally the effects of the different optimisation techniques and development of the search trees were analysed. It was determined that the application of the search optimisation techniques resulted in a overall reduction in the LPM and conflict detection module calls, across all simulations. Therefore, the application of the search optimisation techniques improved the computational efficiency of the algorithm. The development of the search trees were also analysed, to determine whether the horizontal and vertical search trees should be created sequentially or in parallel. A conclusion was reached that although both methods resulted in equivalent final path costs, the development of the trees in parallel ensured that the algorithms ability to derive an initial path to the goal was not effected by the geometry of the scenario.

The adapted RRT* algorithm determined safe near optimal solutions during most of the simulations, showing that the application of the algorithm as a conflict resolution system is possible. However, additional tests are required to ensure the robustness and determine the cause of certain unexpected results. Finally it is

important to note that the algorithm determined simple paths, consisting of a few resolution manoeuvres. Thereby making it possible for a human pilot to track the desired resolution paths.

Chapter 7

Conclusions

7.1 Conclusions

The development of a reliable conflict avoidance system is of paramount importance as the current demand for air travel increases. Increasingly congested airspaces are to be expected in the future, especially surrounding airports. This increase in air travel could result in an increased probability of conflict scenarios. The development of a conflict resolution system that is capable of handling multiple intruders and terrain within an airport environment is required to ensure that potential conflicts are safely avoided. This thesis aimed to develop a conflict resolution system for an aircraft that can:

1. primarily determine a safe flyable path as fast as possible,
2. while additionally utilising any remaining computation time to optimise this path.

Each section in this document explains the development of this system. Justification for the new system was determined early, when analysing existing resolution systems. These systems exhibited a few limitations that the proposed system aimed to remedy. A thorough review of the existing research in the field of path planning and conflict resolution resulted in the selection of sampling-based algorithms, due to their fast execution times and probabilistic completeness. Numerous sampling-based algorithms exist; we selected the RRT* algorithm developed by Karaman and Frazzoli [62]. The RRT* algorithm derives a path to the goal state quickly and additionally has been proven to be asymptotically optimal. This ensures that the algorithm will strive towards an optimal solution. The tree structure utilised by the RRT* simplifies the application of the algorithm to dynamically constrained vehicles as well as dynamic environments. To ensure that the paths derived by the system lie within the host aircraft's envelope of operation, an representative model of the aircraft was derived. The application of the system to a dynamic environment required the modelling of all obstacle aircraft and terrain. Finally changes were made to the RRT* algorithm that resulted in the adapted RRT* proposed in this thesis.

The adapted RRT* is a sampling-based motion planning algorithm that has been adapted for the application as a conflict resolution system. The original RRT* exhibited numerous favourable characteristics, including

probabilistic completeness and asymptotic optimality. The proposed algorithm makes use of these characteristics in conjunction with some additional adaptations to ensure that the algorithm functions effectively when applied as a conflict resolution system. The additional adaptations applied to the algorithm include the development of an aircraft specific LPM module, the application of an effective sampling system and the utilisation of numerous search optimisation techniques. Different LPM functions were developed specifically for the application of the system to an aircraft. This ensured the calculation of dynamically realisable paths in both the horizontal and vertical domains. A dynamic elliptical sampling region in conjunction with rejection sampling guaranteed that the samples generated would promote optimal path calculations, while remaining dynamically realisable. The numerous search optimisation techniques applied to the algorithm reduced the computation time of the system, by limiting the number of function calls through the application of different pruning functions.

The final system was subjected to various statistical tests to determine whether the adapted RRT* algorithm achieved the aims set forth in the problem statement. The algorithm was tested on numerous conflict scenarios, including: the generic two aircraft scenario, a terrain only scenario, a two aircraft with terrain scenario and a multiple aircraft with terrain scenario. Three cost functions based on distance, energy and path deviation were applied during each scenario. These tests were conducted to evaluate the algorithm's performance when applied to dynamic and static environments varying in complexity. In all the scenarios considered the algorithm derived an initial path to the goal state quickly and within few samples. Additionally the geometry of the scenario did not affect the algorithm's ability to quickly derive an initial path, given that a path exists to the goal state. The convergence rate of the algorithm was tested, and both the distance and energy-based cost functions exhibited a high convergence rate to an optimal solution. These cost functions converged to an optimal path within a hundred samples. However, the path deviation method converged to an optimal solution at a slower rate.

Overall the algorithm performed well in all statistical tests applied, and achieved the goals set out in the problem statement. Some unexpected results were recorded, such as the slow convergence rate when applying the path deviation cost function, but this can be attributed to the lack of representative search optimisation techniques. However, additional tests still have to be performed to determine the exact cause. The development of a cost function based sampling region and search optimisation functions should improve the convergence rates and reduce the conflict detection module calls for both the energy and path deviation-based cost functions. The positive results exhibited by the fast initial path calculations coupled with high optimal cost convergence rates show that the application of the algorithm as an aircraft resolution system is viable; however, further tests are required in more complex environments to verify the robustness and safety of the algorithm.

7.2 Future Work

The proposed algorithm developed during the course of this thesis achieves the goals stated in the problem statement. However, additional adaptations can be made that could potentially improve the proposed system. These improvements and adaptations are listed below.

1. LPM module

(a) The addition of more manoeuvres, potentially through the application of a lookup table, can vastly increase the algorithm's ability to determine more optimal paths. The application of additional manoeuvres allows the algorithm to determine dynamically realisable paths to more states within the sampling region, thereby increasing the algorithm's reachability within the search space.

(b) The expansion of the decoupled LPM to a three dimensional system, that is capable of deriving paths consisting of consecutive horizontal and vertical manoeuvres, allows the development of complex paths and ensures that all possible manoeuvre combinations are used to resolve for a conflict. This will increase the size of the search region dramatically, potentially improving the algorithm's capability of deriving paths in very complex scenarios.

(c) The development of an LPM that is capable of handling a constant acceleration assumption will increase the complexity of the developed system. However, this would allow the LPM to plan manoeuvres that are more representative of the aircraft's dynamic constraints as the acceleration of the aircraft has to be taken into account when determining whether paths are indeed flyable.

(d) Creating a manoeuvre set, consisting of manoeuvres that are derived based on the applied cost function. Adapting the development of the manoeuvres to promote an optimal path between two states, based on the applied cost function, could increase the convergence rate of the algorithm and ensure that optimal solutions are determined quickly.

2. Sampling Region

   (a) The application of a three dimensional sampling region required to implement expanded LPM module described above. This ensures that the entire flyable space is utilised when deriving a resolution path.

   (b) The effect of sampling in time could be researched and results compared to the algorithm tested in this thesis.

   (c) The derivation of a cost function specific sampling region. This implies the development of a sampling region shape derived for each specific cost function applied.

3. The proposed conflict resolution algorithm should be integrated with a probabilistic conflict detection algorithm and their combined operation, performance and computational load should be tested experimentally and the results should be analysed. The use of a probabilistic detection module will allow the application of the system to uncertain environments.

4. Adapted RRT*: When connecting a newly sampled state to the search tree the algorithm could make use of a set connection radius, resulting in a spherical LPM connection range around each newly sampled state. This set maximum connection range could reduce the effect that the size of the tree has on the computational complexity of the algorithm. This method was used in the RRT*, proposed by Karaman and Frazzoli [62]. The application of it to the algorithm could vastly reduce the number of LPM and conflict detection module calls, thereby increasing the computation speed of the algorithm.

5. Search Optimisation Techniques: The currently developed search optimisation techniques were created for the distance-based cost function. However, the methods did show positive results when applied to the energy and path deviation-based cost functions. The application of cost function specific search optimisation techniques should yield better results and the development of these methods could further improve the effectiveness of the system.

6. A combined cost function should be investigated. The individual cost functions each have their own favourable attributes. A combination of these cost functions could exploit these attributes and produce better optimal paths.

7. Additional Tests

   (a) The current tests assumed that the algorithm is active on only the host aircraft. A test can be conducted to determine the validity of the algorithm if the system is active on all aircraft in the environment.

   (b) A highly cluttered environment can be tested to ensure that the algorithm is robust.

   (c) Tests have to be done to explain the slow cost convergence rate of the path deviation-based cost function.

Appendix A

Search Algorithms

In order to extract the optimal path determined by the sampling-based algorithm, we require a method to search through the graphs generated. This can be accomplished by applying any form of search algorithm to the graph structure created. Application of a search algorithm to the path planning problem requires that it is systematic [10]. Being systematic means that the search algorithm will visit each reachable vertex in a finite graph. This enables it to declare whether a solution exists within a certain finite time. Essentially, if it can be ensured that the algorithm does not cycle through previously visited vertices, thereby removing redundant exploration, the algorithm can be considered to be systematic.

Most search algorithms can be grouped into forward, backward and bidirectional searches. The most common is the forward search which contains: the breadth first search, depth first search, Dijkstra's algorithm, A-star, best-first search and iterative deepening. The difference between the groups is in the method of searching. The forward search method will start at the initial vertex and search towards the goal. The backward search does the exact opposite, starting at the goal and moving back towards the initial vertex. Bidirectional searches from the start at both goal and initial vertices and moves towards each other and ultimately link in the centre. Of all the search algorithms listed above, we have identified two promising candidates: Dijkstra's algorithm and the A-star search. These algorithms determine the optimum path efficiently regardless of the shape of the graph, while remaining systematic [10].

A.1 Dijkstra's Algorithm

The Dijkstra search [87] algorithm determines the optimum path for single source graphs. This method of search algorithm can be considered to be a special form of dynamic programming. Here follows an explanation by LaValle of the execution and functioning of the algorithm [10].

A prerequisite for the implementation of a sorting algorithm in general is that each edge, $e \in E$, is associated with an execution cost. This cost can be determined through any predefined arbitrary positive function $f^+(e)$ (see Section 5.7 on the different cost functions used). The state space representation of this cost function can

be represented as $f^+(v, u)$, where u is the manoeuvre or action executed from state v. The algorithm makes use of a priority queue (Q) to sort the edges using the following cost to come formula $C : V \to [0, \infty]$, where $v \in V$. This allows for faster computation when searching through the graph. For each state in the graph there exists an optimal cost to come value $C^*(v)$. This cost is obtained by comparing all possible cumulative path costs to that state and selecting the path that results in the lowest cost.

The algorithm starts at $C^*(v_I) = 0$, where state $v_0 = q_{init}$. Any state linked to v_0 through an edge is selected and its cost is determined: $C(v') = C^*(v) + f^+(e)$, where e is the edge from v to v'. Note that $C(v')$ represents the best cost to come so far and not the optimal cost. If the priority queue Q already contains a cost for v', then a check is made to determine which cost is lower. The lower cost is selected and the priority queue is updated and reordered. When all the paths have been tested to state v it is removed from Q and then $C(v)$ becomes $C^*(v)$. This process is applied to all the states in the graph to determine the optimal cost to each individual state. From this the cost to the final state can be determined and the optimal path extracted.

A.2 A-star

The Dijkstra algorithm can be classified as a special case of the A-star search [9]. The A-star search determines the optimal path in exactly the same fashion as the Dijkstra search, explained above. The main difference is in the method used to sort the priority queue Q [10]. This method attempts to apply a certain heuristic underestimate of the cost to go in order to reduce the number of states visited before an optimal path is found. The heuristic underestimate must be as close to the optimal cost as possible, but must ensure that it does not exceed the optimal cost. The queue is therefore sorted by $C^*(v') + \hat{C}^*(v')$, where $\hat{C}^*(v)$ is the underestimated optimal path for all $v \in V$. The Dijkstra algorithm is the special case where the heuristic underestimate is selected as $\hat{C}^*(v) = 0$ for all states in V. The efficiency and effectiveness of the A-star search is closely coupled to the selection of the heuristic underestimate. Where bad estimates result in slower execution times and possibly suboptimal paths, it can be complicated to determine an optimal heuristic estimate and overestimation is always potential problem.

Appendix B

Calculation of Dubins Curves

The calculations required to determine the viable Dubins curves for all implementations described in Section 5.6.2 can be reduced to the calculation of the intersection points between two circles. Assume that we have two arbitrary circles, in the two dimensional horizontal space, centred at $O_1 = (a, b)$ and $O_2 = (c, d)$, as seen in Figure B.1.

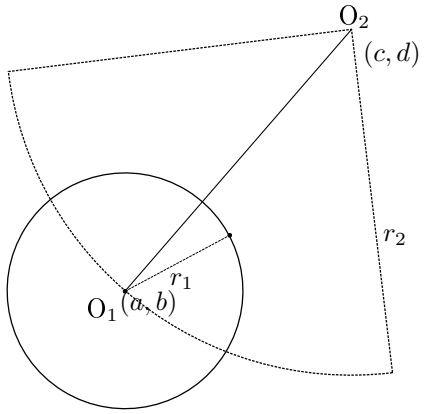


Figure B.1: Two arbitrary intersecting circles

The two circles depicted in Figure B.1 can be represented by:

$$(x-a)^2 + (y-b)^2 = r_1^2 \tag{B.1}$$

$$(x-c)^2 + (y-d)^2 = r_2^2, \tag{B.2}$$

where r_1 and r_2 represent the radii of the two circles. Now all that is required is solving the simultaneous equations between the two circles and determining their corresponding intersection points. The resultant formulae for the x-coordinate is:

$$x = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}, \tag{B.3}$$

where

$$A = \left(\frac{c-a}{d-b}\right)^2 + 1 \tag{B.4}$$

$$B = \frac{(c-a)\left[(c-a)^2 + (d-b)^2 + r_2^2 - r_1^2\right]}{(d-b)^2} \tag{B.5}$$

$$C = \left[\frac{(c-a)^2 + (d-b)^2 + r_2^2 - r_1^2}{2(d-b)}\right] - r_1^2. \tag{B.6}$$

From Equation B.3 and B.1 we can determine the required x-coordinate and the corresponding y-coordinate. Now the intersection points between a tangent line touching the circle O_1 and passing through position (c, d) has been determined. This calculation can be used to determine the intersection points for all the Dubins curve methods used.

Appendix C

Probability of Rejection Calculations

The rhombus shaped acceptance zone determined for the vertical LPM in Section 5.8.2 requires, for efficient execution, that the probability of accepting a sample is larger than rejecting it. This probability can be determined by determining the chance of sampling within the acceptance zone versus the probability of sampling outside it, assuming an elliptical sampling region. These probabilities are directly proportional to the areas of the corresponding zones, due to the application of a uniform sampling strategy. Therefore we have to determine the area of the acceptance rhombus as well as the area of the rejection zone.

The area of the acceptance rhombus remains constant throughout the conflict resolution algorithm's functioning, while the elliptical sampling region is dynamically adapted (see Section 5.8.3). The probability of rejection will be a maximum when the rejection zone's area is the largest, therefore the original sampling size is used to determine the worst case rejection probability.

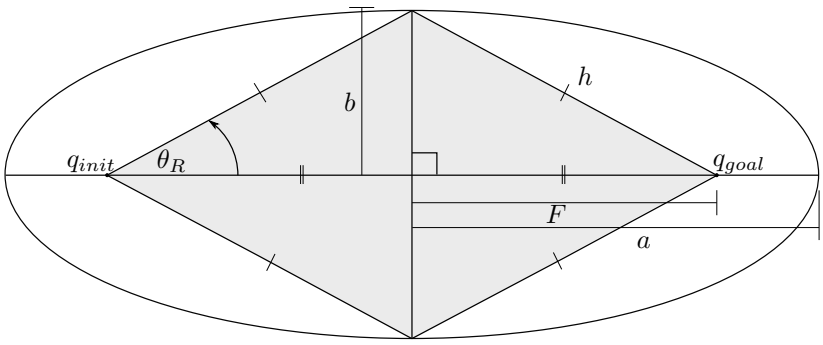


Figure C.1: The rejection and acceptance zones within an elliptical sampling region

The areas of the ellipse and rhombus (A_{ellip}, A_{rhom}), depicted in Figure C.1, can be described by:

$$A_{ellip} = \pi ab \tag{C.1}$$

$$\begin{aligned} A_{rhom} &= \frac{d_1 d_2}{2} \\ &= 2bF, \end{aligned} \tag{C.2}$$

where d_1, d_2 are the diagonals of the rhombus and therefore translate to $d_1 = 2b$ and $d_2 = 2F$. We also know from Equation 5.4 that $F = \sqrt{a^2 - b^2}$.

Efficient rejection sampling requires that the acceptance zone (A_{rhom}) always remains larger than the rejection zone ($A_{ellip} - A_{rhom}$), therefore:

$$\begin{aligned} A_{ellip} - A_{rhom} &\leq A_{rhom} \\ A_{ellip} &\leq 2A_{rhom} \\ \pi ab &\leq 4bF \end{aligned} \tag{C.3}$$

By substituting the formulation of F from Equation 5.4 in and solving for b/a we determined:

$$\begin{aligned} \frac{\pi a}{4} &\leq \sqrt{a^2 - b^2} \\ b^2 &\leq a^2 \left(1 - \left(\frac{\pi}{4}\right)^2\right) \\ \therefore \frac{b}{a} &\leq \sqrt{1 - \left(\frac{\pi}{4}\right)^2} \\ &\leq 0.61899 \end{aligned} \tag{C.4}$$

Equation C.4 depicts the required ratio between the minor and major axes of the ellipse to ensure that the acceptance region is statistically favoured. From this we can determine the required flight path angle θ that will ensure that the acceptance region is favoured:

$$\theta_R = \arctan \frac{b}{a} \tag{C.5}$$

$$\therefore \theta_R \leq \arctan 0.61899 \tag{C.6}$$

$$\leq 31.75713^\circ \tag{C.7}$$

Equation C.7 states that as long as the maximum climb rate of the aircraft does not result in a flight path angle greater than 31.75713° then rejection sampling will always have a larger probability of accepting a sample.

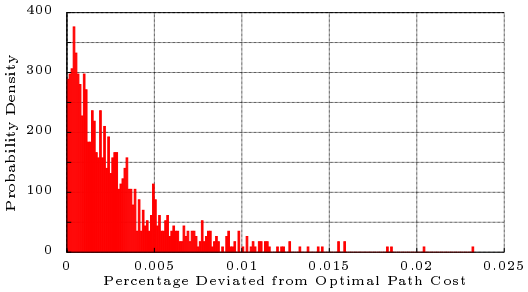
Appendix D

Results Data

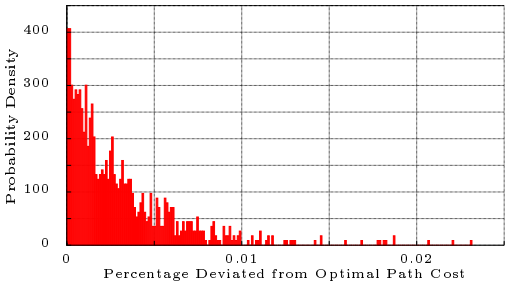
This section contains all additional data and figures relevant to the Simulations and Results chapter.

Table D.1: Statistical results depicting optimal path characteristics, determined from 1000 simulations at 10 iterations for each scenario.

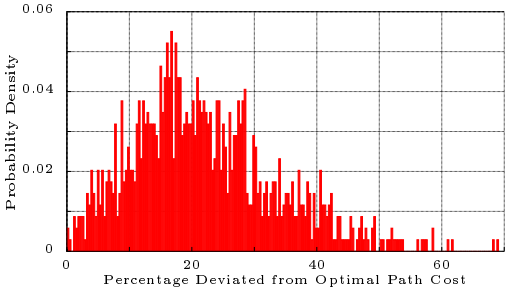
Cost Function	**Average Cost ($\bar{c}$)**	
Generic Two Aircraft	% of c^*	$\sigma_{\%}$
Distance	100.01097	0.00572
Energy	100.00637	0.00652
Path Deviation	666.8372	34.5147
Terrain only		
Distance	100.3071	4.4007
Energy	100.531	5.3018
Path Deviation	286.4881	122.52
Two Aircraft, with Terrain		
Distance	102.8431	13.3933
Energy	104.9057	15.4234
Path Deviation	753.3647	76.8459
Multiple Aircraft, with Terrain		
Distance	107.6388	314.116
Energy	111.5303	208.734
Path Deviation	2255.8105	69.509



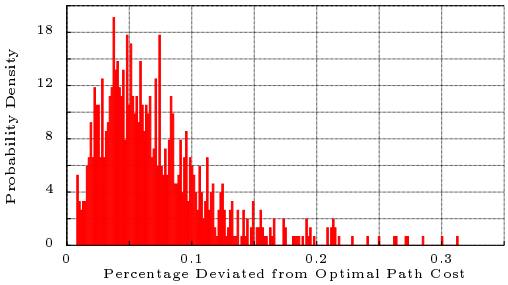
(a) Terrain only – Distance-Based



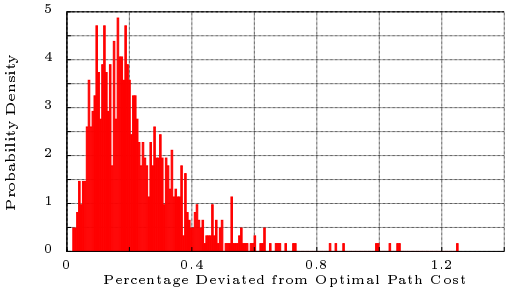
(b) Terrain only – Energy-Based



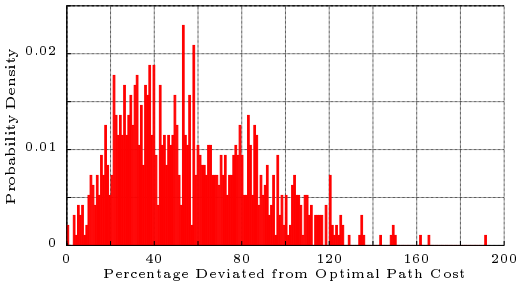
(c) Terrain only – Path Deviation-Based



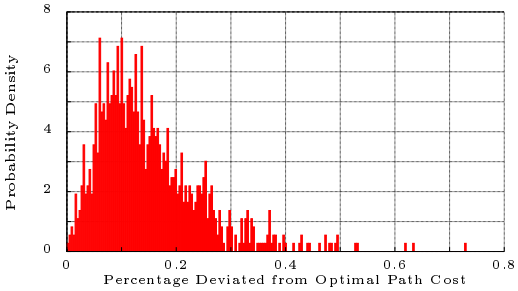
(d) Single Aircraft with Terrain – Distance-Based



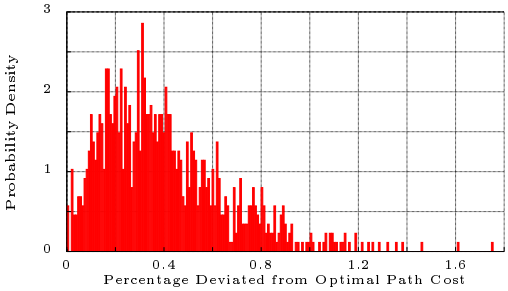
(e) Single Aircraft with Terrain – Energy-Based



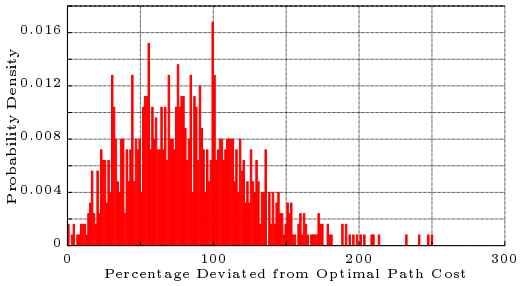
(f) Single Aircraft with Terrain – Path Deviation-Based



(g) Multiple Aircraft with Terrain – Distance-Based



(h) Multiple Aircraft with Terrain – Energy-Based



(i) Multiple Aircraft with Terrain – Path Deviation-Based

Figure D.1: Histograms of the optimal path cost PDF, for the all scenarios at $I_S = 100$, normalised as a percentage from the optimal solution determined at $I_S = 10000$.

Bibliography

[1] J. Hoekstra, R. van Gent, and R. Ruigrok, "Conceptual design of free flight with airborne separation assurance," Nationaal Lucht- en Ruimtevaartlaboratorium, National Aerospace Laboratory NLR, June 1998.

[2] J. K. Kuchar and L. C. Yang, "Survey of conflict detection and resolution modeling methods," in *Proccedings of AIAA Guidance, Navigaton and Control Conference*, 1997.

[3] Federal Aviation Administration (FAA), "FAA flight standards pilot outreach program," in *Traffic Alert and Collision Avoidance System (TCAS)*, June 2012.

[4] M. S. Eby, "A self-organizational approach for resolving air traffic conflict," *The Lincoln Laboratory Journal*, vol. 7, no. 2, pp. 239 – 253, 1994.

[5] I. Peddle, "Advanced automation 833." Stellenbosch University: Lecture Notes, 2013.

[6] P. Hachenberger, "Cgal 4.4 - 3d: Minkowski sum of polyhedra," March 2014.

[7] C. E. van Daalen, "Conflict detection and resolution for autonomous vehicles," *PhD dissertation, Stellenbosch University*, March 2010.

[8] R. Diestel, *Graph Theory*. Graduate Texts in Mathematics, Springer Berlin Heidelberg, 2010.

[9] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents series)*. The MIT Press, June 2005.

[10] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[11] T. R. Yechout, *Introduction to aircraft flight mechanics*. AIAA, 2003.

[12] G. Chaloulos, J. Lygeros, I. Roussos, K. Kyriakopoulos, E. Siva, A. Lecchini-Visintini, and P. Cásek, "Comparative study of conflict resolution methods." iFly Deliverable D5.1, December 2007. `http://ifly.nlr.nl/publications.html`.

[13] Federal Aviation Administration (FAA), *Pilot's Handbook of Aeronautical Knowledge*, 2008.

[14] E. G. Cowley, "Kinodynamic planning for a fixed-wing aircraft in dynamic, cluttered environments: A local planning method using implicitly-defined motion primitives," *Masters Thesis, Stellenbosch University*, March 2013.

[15] International Civil Aviation Organization (ICAO), "Annual passenger total approaches 3 billion according to ICAO 2012 air transport results." ICAO News Release, December 2012.

[16] L. J. Pienaar, "Probabilistic conflict detection for commercial aircraft near airports," *Masters Thesis, Stellenbosch University*, 2014.

[17] H.-H. Nguyen, "Survey of coordination of en route air traffic conflicts resolution modelling methods," *RIVF*, pp. 43–48, 2003.

[18] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000.

[19] International Civil Aviation Organization (ICAO), "Air traffic management," in *Procedures for Air Navigation Services*, 2007.

[20] J. K. Kuchar and L. C. Yang, "Using intent information in probabilistic conflict analysis," *AIAA Guidance, Navigation, and Control Conference*, pp. 797 – 860, 8 1998.

[21] C. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution manoeuvres," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 2, pp. 110–120, Jun 2001.

[22] J. Krozel and M. Peters, "Strategic conflict and detection and resolution and for free flight," in *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, vol. 2, pp. 1822–1828, IEEE, December 1997.

[23] Honeywell, *KTA870 / KMH880 Pilot's Guide: Traffic Advisory System / Multi-Hazard Awareness System*, 2005.

[24] Federal Aviation Administration (FAA), "Air carrier operational approval and use of tcas ii," in *Advisory Circular 120-55C*, U.S. Department of Transportation, 2013.

[25] International Civil Aviation Organization (ICAO), "Airborne collision avoidance system (ACAS) manual," 2006.

[26] Y. Mechqrane and E. H. Bouyakhf, "Systematic vs. non-systematic search for 3d aircraft conflict resolution," *Journal of Intelligent Learning Systems and Applications*, vol. 4, p. 223, 2012.

[27] The European Organisation for the Safety of Air Navigation (EUROCONTROL), *ACAS II Guide: Airborne Collision Avoidance System II (incorporating version 7.1)*, 1 ed., January 2012.

[28] International Civil Aviation Organization (ICAO), "Rules of the air," in *Annex 2 to the Convention on International Civil Aviation*, November 2005.

[29] D. W. Burgess, S. I. Altman, and M. L. Wood, "Tcas: Maneuvering aircraft in the horizontal plane," *Lincoln Lab. J.*, vol. 7, pp. 295–312, Sept. 1995.

[30] Federal Aviation Administration (FAA), "Evaluation of candidate functions for traffic alert and collision avoidance system ii (tcas ii) on unmanned aircraft system (uas) ," in *FAA/AFS-407*, 4 2011.

[31] Honeywell, *Pilots Guide: Enahnced Ground Proximity Warning System (EGPWS) and Flight Safety Functions TSO C151b Class A TAWS*, March 2013.

[32] M. Eby and W. I. Kelly, "Free flight separation assurance using distributed algorithms," in *Aerospace Conference, 1999. Proceedings. 1999 IEEE*, vol. 2, pp. 429–441, 1999.

[33] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," tech. rep., DTIC Document, 1993.

[34] T. M. Jimmy Krozel and G. Hunter, "Free flight conflict detection and resolution analysis," *Guidance, Navigation, and Control Conference*, July 1996.

[35] C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: a study in multiagent hybrid systems," *Automatic Control, IEEE Transactions on*, vol. 43, pp. 509–521, Apr 1998.

[36] J. P. Wangermann and R. F. Stengel, "Principled negotiation between intelligent agents: a model for air traffic management," *Artificial Intelligence in Engineering*, vol. 12, no. 3, pp. 177 – 187, 1998.

[37] J. P. Wangermann and R. F. Stengel, "Optimization and coordination of multiagent systems using principled negotiation," *Journal of Guidance, Control, and Dynamics*, pp. 43–50, 1999.

[38] N. Durand and J.-M. Alliot, "Optimal resolution of en route conflicts," in *In 1st Air Traffic Management R&D Seminar*, 1997.

[39] W. Love, "TCAS III - Bringing operational compatibility to airborne collision avoidance," *Digital Avionics Systems Conference.*, October 1988.

[40] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, pp. 566–580, August 1996.

[41] Airports Council International ACI, "Media release: Aci releases its 2012 world airport traffic report," 2012.

[42] Civil Aviation Athority (CAA), "Enr 1.2 - visual flight rules," in *United Kingdom Integrated Aeronautical Information Package (IAIP)*, December 2012. http://www.ead.eurocontrol.int...

[43] Civil Aviation Athority (CAA), "Enr 1.7 - altimeter setting procedures," in *United Kingdom Integrated Aeronautical Information Package (IAIP)*, May 2013. http://www.ead.eurocontrol.int...

[44] Federal Aviation Administration (FAA), *Instrument Flying Handbook: FAA-H-8083-15B*. FAA Handbooks, Aviation Supplies & Academics, Inc., January 2013.

[45] International Civil Aviation Organization (ICAO), "Air traffic services," in *Annex 11 to the Convention on International Civil Aviation*, November 2005.

[46] Civil Aviation Athority (CAA), "Enr 1.1 - general rules and procedures," in *United Kingdom Integrated Aeronautical Information Package (IAIP)*, Jan 2013. http://www.ead.eurocontrol.int...

[47] G. Atul, L. Rezawana Islam, and C. Tonoy, "Evolution of aircraft flight control system and fly-by-light flight control system," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, pp. 60–64, 2013.

[48] C. R. Spitzer and C. Spitzer, *Digital avionics handbook*. CRC Press, 2000.

[49] C. Favre, "Fly-by-wire for commercial aircraft: the airbus experience," *International Journal of Control*, vol. 59, no. 1, pp. 139–157, 1994.

[50] Federal Aviation Administration (FAA), *Advanced Avionics Handbook FAA-H-8083-6*. U.S. Government Printing Office, 2009.

[51] N. H. McClamroch, *Steady aircraft flight and performance*. Princeton University Press, 2011.

[52] Federal Aviation Administration (FAA), "United states standard for performance based navigation (pbn) instrument procedure design," vol. 6, 2012.

[53] B. Wainwright, J. C. Rockliff, J. Cashman, T. Melody, D. Carbaugh, M. Cariker, and D. Forsyth, "Aerodynamic principles of large-airplane upsets," *Airbus and Boeing white paper*, 2002.

[54] S. Skiena, *The Algorithm Design Manual*. Springer Science & Business Media, 2009.

[55] D. Mount, "Cmsc 754 computational geometry." University of Maryland Lecture Notes, 2002.

[56] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[57] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1056–1062, 2008.

[58] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, pp. 46–57, June 1989.

[59] F. Tanedo, "Notes on non-holonomic constraints." Cornell University Lecture Notes, 2013.

[60] M. T. Mason, "Nonholonomic constraint." Carnegie Mellon University Lecture Notes, 2013.

[61] J. I. Neimark and Nikolaĭ, *Dynamics of Nonholonomic Systems*, vol. 33 of *Translations of mathematical monographs*. American Mathematical Society, 1972.

[62] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics: Science and Systems Conference*, vol. abs/1005.0416, 2010.

[63] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[64] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," in *Technical Report 98-11, Computer Science Department, Iowa State University, October*, 1998.

[65] M. H. Overmars, "A random approach to motion planning," *RUU-CS*, pp. 92–32, 1992.

[66] L. E. Kavraki, *Random Networks in Configuration Space for Fast Path Planning*. PhD thesis, Stanford, CA, USA, 1995. UMI Order No. GAX95-16854.

[67] P. Svestka, M. H. Overmars, M. H. Overmars, and M. H. Overmars, "Motion planning for car-like robots using a probabilistic learning approach," *Int. J. of Robotics Research*, vol. 16, 1995.

[68] L. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 166 – 171, 1998.

[69] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration for fast path planning," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 2138–2145 vol.3, May 1994.

[70] K. L. Cheung and A. W.-C. Fu, "Enhanced nearest neighbour search on the r-tree," *ACM SIGMOD Record*, vol. 27, no. 3, pp. 16–21, 1998.

[71] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[72] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 2, pp. 995–1001 vol.2, 2000.

[73] P. Pharpatara, B. Hérissé, R. Pepy, and Y. Bestaoui, "Sampling-based path planning: a new tool for missile guidance," *19th IFAC Symposium on Automatic Control in Aerospace*, pp. 131–136, 2013.

[74] E. Mazer, J. M. Ahuactzin, and P. Bessiere, "The ariadne's clew algorithm," *Journal of Artificial Intelligence Research (JAIR)*, vol. 9, pp. 295–316, 1998.

[75] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 3, pp. 2719–2726, IEEE, 1997.

[76] S. Carpin and G. Pillonetto, "Motion planning using adaptive random walks," *Robotics, IEEE Transactions on*, vol. 21, no. 1, pp. 129–136, 2005.

[77] M. Hlynka and D. Loach, "Generating uniform random points in a regular n sided polygon." Dept. of Mathematics and Statistics, University of Windsor, Windsor, Ontario, Canada N9B 3P4, July 2014. Lecture Notes.

[78] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA Journal of Guidance, Control and Dynamics*, vol. 25, pp. 116–129, 2002.

[79] L. E. Dubins, "On plane curves with curvature.," *Pacific Journal of Mathematics*, vol. 11, no. 2, pp. 471–481, 1961.

[80] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *Robotics, IEEE Transactions on*, vol. 21, no. 6, pp. 1077–1091, 2005.

[81] M. J. Kochenderfer, J. P. Chryssanthacopoulos, L. P. Kaelbling, and T. Lozano-Pérez, "Model-based optimization of airborne collision avoidance logic," tech. rep., DTIC Document, 2010.

[82] J. D. Anderson, *Aircraft Performance and Design*. McGraw-Hill Education (India) Pvt Limited, 1999.

[83] A. Filippone, *Advanced Aircraft Flight Performance*. Advanced Aircraft Flight Performance, Cambridge University Press, 2012.

[84] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3, pp. 2383–2388, IEEE, 2002.

[85] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5, pp. 4350–4355, IEEE, 2004.

[86] M. Kalos and P. Whitlock, *Monte Carlo Methods*. Monte Carlo Methods, Wiley, 1986.

[87] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.