CRANFIELD UNIVERSITY

**Chi Kin Lai**

# A NOVEL COLLISION AVOIDANCE LOGIC FOR UNMANNED AERIAL VEHICLES USING REAL-TIME TRAJECTORY PLANNING

SCHOOL OF ENGINEERING

PhD THESIS

CRANFIELD UNIVERSITY

SCHOOL OF ENGINEERING

PhD THESIS

Academic Year 2013-2014

Chi Kin Lai

# A Novel Collision Avoidance Logic for Unmanned Aerial Vehicles Using Real-Time Trajectory Planning

Supervisor: Dr James F. Whidborne

October 2014

©Cranfield University 2014. All rights reserved. No part of this publication may be reproduced without the written permission of the copyright owner.

*For my family and especially my fiancée,*

*Hoi Lam Cheong,*

*who always supports.*

*With special thanks to,*

*Ip Shing Long and Ana Leong,*

*who believed.*

# Acknowledgements

I would first like to thank my supervisor, Dr James Whidborne, for his patience to listen to my scattered, and sometime unclear, thoughts and more importantly to help me understanding those thoughts better by asking the right questions.

I would also like to thank Dr Darren Ansell for his useful advices on my work and the arrangement of my placement in BAE Systems. Particular thanks go to Richard Freeman, Robert Dixon, Dr Charles Patchett, Rod Buchanan, Paul Marsland and Fish Lee from BAE Systems for their support and valuable discussions during my placement.

I am also thankful to the members of Dynamics, Simulation and Control Group for all those random but sometime inspiring discussions over the tea time. Special thanks go to Dr Mudassir Lone who not only shared his experience and work but also offered countless helps during my PhD, Dr Rick Drury and Quintain Mcenteggart who shared their own experiences and knowledge in trajectory optimization, Dr Peter Thomas who shared his vehicle model, and Dr Pierre-Daniel Jameson, Dr King Tin Leung, Dr Sunan Chumalee, Dr Neil Panchal and Dr Ramey Jamil who helped me out with starting a PhD.

I am also grateful to BAE Systems and EPSRC for the bursary and support under an Industrial CASE award.

# Abstract

An effective collision avoidance logic should prevent collision without excessive alerting. This requirement would be even more stringent for an automatic collision avoidance logic, which is probably required by Unmanned Aerial Vehicles to mitigate the impact of delayed or lost link issues. In order to improve the safety performance and reduce the frequency of false alarms, this thesis proposes a novel collision avoidance logic based on the three-layer architecture and a real-time trajectory planning method. The aim of this thesis is to develop a real-time trajectory planning algorithm for the proposed collision avoidance logic and to determine the integrated logic's feasibility, merits and limitations for practical applications.

To develop the trajectory planning algorithm, an optimal control problem is formulated and an inverse-dynamic direct method along with a two stage, derivative-free pattern search method is used as the solution approach. The developed algorithm is able to take into account the flyability of three dimensional manoeuvres, the robustness to the intruder state uncertainty and the field-of-regard restriction of surveillance sensors. The testing results show that the standalone executable of the algorithm is able to provide a flyable avoidance trajectory with a maximum computation time less than 0.5 seconds.

To evaluate the performance of the proposed logic, an evaluation framework for Monte Carlo simulations and a baseline approach for comparison are constructed. Based on five Monte Carlo simulation experiments, it is found that the proposed logic should be feasible as 1) it is able to achieve an update rate of 2Hz, 2) its safety performance is comparable with a reference requirement from another initial feasibility study, and 3) despite a 0.5 seconds computation latency, it outperforms the baseline approach in terms of safety performance and robustness to sensor and feedback error.

# Contents

**Contents** vii

**List of Figures** xiii

**List of Tables** xxi

**Nomenclature** xxv

**Acronyms** xxxiii

**1 Introduction** 1

1.1 Motivations . . . . . 1
1.2 Challenges . . . . . 3
1.3 Overview of the Proposed Approach . . . . . 4
1.4 Aim and Objectives . . . . . 5
1.5 Organization and Highlights of this Thesis . . . . . 6

**2 Background and Literature Review** 11

2.1 Introduction . . . . . 11
2.2 Conflict Management . . . . . 11
2.2.1 Layered Approach . . . . . 11
2.2.2 Safety Events . . . . . 12
2.2.3 Risk Mitigation Mechanisms . . . . . 13
2.3 Sense and Avoid Systems . . . . . 14
2.3.1 A Safety-Analysis Development Process . . . . . 15
2.3.2 Intended Functions and Sub-Functions . . . . . 17

2.3.3 Functional Architecture . . . . . 18
2.4 Collision Avoidance Logics . . . . . 20
2.4.1 Conflict Detection and Resolution . . . . . 20
2.4.2 Challenges and State-of-the-Arts . . . . . 22
2.4.3 Problem Scope . . . . . 25
2.5 Trajectory Planning . . . . . 26
2.5.1 Trajectory Planning rather than Path Planning . . . . . 26
2.5.2 Optimal Control Approach . . . . . 26
2.6 Summary . . . . . 27

**3 Modelling and Simulation Framework** **29**
3.1 Introduction . . . . . 29
3.2 Encounter Model . . . . . 30
3.2.1 Encounter Modelling . . . . . 31
3.2.2 ICAO Standard Encounter Model . . . . . 32
3.2.3 Encounter Generation . . . . . 35
3.2.4 Encounter Simulation . . . . . 38
3.3 Intruder Model . . . . . 38
3.3.1 Manoeuvre Uncertainty . . . . . 39
3.3.2 Intruder Limitations . . . . . 39
3.3.3 State Equation . . . . . 40
3.4 Aircraft System Model . . . . . 40
3.4.1 Aircraft Dynamics . . . . . 41
3.4.2 Navigation System . . . . . 45
3.4.3 Manoeuvre Autopilot . . . . . 46
3.5 Surveillance System Model . . . . . 51
3.5.1 Actual Relative Motion . . . . . 53
3.5.2 Field of Regard . . . . . 54
3.5.3 Track Maintenance . . . . . 56
3.5.4 Measurement Error . . . . . 58
3.5.5 Target Tracking . . . . . 59
3.6 Collision Avoidance Logic Model . . . . . 60
3.6.1 Conflict Detection . . . . . 61

3.6.2 Conflict Resolution . . . . . 63
3.6.3 Conflict Monitor . . . . . 65
3.7 Summary . . . . . 66

**4 Evaluation and Analysis Framework** **71**
4.1 Introduction . . . . . 71
4.2 Typical Performance Metrics . . . . . 72
4.2.1 Possible Outcomes of an Encouter Scenario . . . . . 72
4.2.2 Metrics Definitions . . . . . 73
4.3 Additional Metrics for Non-Cooperative Resolution . . . . . 74
4.3.1 Basic Concepts . . . . . 74
4.3.2 Metrics Definitions . . . . . 76
4.4 Two Simulation Experiments . . . . . 77
4.4.1 Experiment Setting . . . . . 77
4.4.2 Establishment of the Baseline Performance . . . . . 79
4.4.3 Investigation of Field-of-Regard Restriction Effect . . . . . 82
4.5 Summary . . . . . 84

**5 Trajectory Planning Algorithm Development** **87**
5.1 Introduction . . . . . 87
5.2 Problem Formulation . . . . . 88
5.2.1 Differential Constraints . . . . . 89
5.2.2 Trajectory Constraints . . . . . 90
5.2.3 Boundary Conditions . . . . . 93
5.2.4 Cost Functions . . . . . 95
5.3 Problem Transcription . . . . . 97
5.3.1 Differential Constraint Removal . . . . . 97
5.3.2 Parameterization . . . . . 100
5.3.3 Boundary Conditions Satisfaction . . . . . 102
5.3.4 Trajectory Discretization . . . . . 103
5.4 Problem Solution . . . . . 105
5.4.1 Solution Space . . . . . 105
5.4.2 Problem Scaling . . . . . 108

5.4.3 Penalty Function . . . . . 110
5.4.4 Optimization . . . . . 111
5.5 Main Results . . . . . 112
5.5.1 Experiment Settings . . . . . 112
5.5.2 Result Statistics . . . . . 114
5.5.3 Example Trajectories . . . . . 116
5.6 Summary . . . . . 118

**6 System Integration and Performance Evaluation 121**
6.1 Introduction . . . . . 121
6.2 Integration of Trajectory Planning . . . . . 122
6.2.1 Trajectory Planner . . . . . 122
6.2.2 Trajectory Manager . . . . . 123
6.2.3 Trajectory Tracker . . . . . 124
6.3 Evaluation of the Proposed Approach . . . . . 128
6.3.1 Safety Performance Evaluation under the Ideal Condition . . . 128
6.3.2 Robustness Analysis on Sensor Noise . . . . . 131
6.3.3 Evaluation of the Field-of-Regard-Restriction Effect . . . . . 132
6.4 Discussions with Example Encounters . . . . . 136
6.4.1 Ineffectiveness of the Emergency Strategy . . . . . 136
6.4.2 Conservativeness in Uncertainty Propagation . . . . . 136
6.4.3 Parallel-Track Results . . . . . 139
6.4.4 Field-of-Regard-Restriction Effects . . . . . 139
6.5 Summary . . . . . 140

**7 Conclusions 143**
7.1 Feasibiliy, Mertis and Limitations . . . . . 143
7.2 Contributions to Knowledge . . . . . 147
7.3 Dissemination of Results . . . . . 148
7.4 Recommendations for Future Research . . . . . 149

**A Notation 156**
A.1 Definitions and Typefaces of Mathematical Objects . . . . . 156

**B Aircraft System Model** **162**
B.1 Aircraft Performance Data . . . . . 162
B.2 Verification Examples . . . . . 162

**C Trajectory Planning Algorithm Implementation Details** **171**
C.1 Hooke-Jeeves Algorithm . . . . . 171
C.2 Algorithm Parameters . . . . . 171
C.3 Example Trajectories for the Infeasible Solutions . . . . . 174

**References** **179**

# List of Figures


1.1 The notional explanation of the desire to use an avoidance manoeuvre of higher performance. . . . . 4

1.2 Architecture of the proposed collision avoidance logics for Sense and Avoids (SAAs) systems. The grey functional blocks highlight the focus of this thesis. More details can be found in §2.3.3 and §6.2 for the architecture and in Chapter 5 for the trajectory planning method. . . . 5

1.3 Main features of the proposed trajectory planning method, more details can be found in Chapter 5. . . . . 5

1.4 Concept of the proposed trajectory planning method of high performance manoeuvres for collision avoidance problems. . . . . 5

1.5 Overview of the organization, deliverables and target domains of contributions of this thesis, see §7.2 for more details. . . . . 6

2.1 Overview of Chapter 2, containing the context of the literature and their relationships to the scope of this thesis. . . . . 11

2.2 The safety layer models illustrating the layered approach used for collision avoidance in the Air Traffic Management (ATM) system. . . . . 12

2.3 Risk mitigation mechanisms grouping into three safety layers. The scope of SAA is also highlighted. . . . . 14

2.4 Safety-analysis methodology for the development of SAA systems. . . 15

2.5 The process to calculate the system risk of collision, adapted from Raynaud & Arino [2006]. . . . . 15

2.6 SAA Capability Sub-Functions, reproduced according to [FAA-Sponsored Workshop, 2009, as cited in [George, 2012]]. . . . . 18

2.7 The functional architecture of a generic SAA system, for autonomous Unmanned Aircraft Systemss (UASs), summarized from Hutchings *et al.* [2007]; Patchett & Ansell [2010]; Dixon [2011]. . . . . 18
2.8 The necessary elements of a conflict detection and resolution (CDR) process, which is the underpinning principle for all collision avoidance logics. . . . . 22
2.9 The relationship between the methods (for intruder state projections and conflcit resolution) used in this thesis to the existing methods in the literature. . . . . 22
2.10 Comparison of the development and usage processes of two generations of Airborne Collision Avoidance System (ACAS), based on Kochenderfer *et al.* [2011]. . . . . 25

3.1 Modelling and Simulation Framework and Overview of Chapter 3. . . 30
3.2 The evolution of the existing encounter models, adapted from Kochenderfer *et al.* [2008a]. . . . . 30
3.3 The outline of §3.2, showing the general process to construct and use an encounter model. . . . . 31
3.4 An example of the model structure of encounter models, using a Bayesian network. . . . . 32
3.5 An example encounter illustrating the encounter window; and the encounter characteristics of time of closest approach ($tca$), approach angle ($\Theta_{app}$), horizontal miss distance ($HMD$), vertical miss distance ($VMD$), and encounter altitude ($H_{enc}$). . . . . 33
3.6 An example encounter to illustrate the model variables that specify the trajectory characteristics. . . . . 35
3.7 Establishment of the Navigation coordinate system and initialization of the aircraft trajectory construction process. . . . . 37
3.8 Intruder model architecture. . . . . 38
3.9 Overview of a configurable aircraft system model. . . . . 41
3.10 Parameter definitions of the aircraft point-mass model in the navigation coordinate system $\mathsf{N}$, adapted from Menon *et al.* [1999] . . . . . 42

3.11 The control scheme for the manoeuvre autopilot, which is based on Nonlinear Dynamic Inversion technique. . . . . 47
3.12 The signal flow diagram for the Manoeuvre Autopilot. . . . . 48
3.13 The Block Diagram for the Desired Dynamics Controller. . . . . 50
3.14 Architecture of surveillance system model, showing the generation of four types of test cases: 1. *Ideal*; 2. *Noisy*; 3. *Field of Regard (FOR) restricted*; 4. *Realistic*. . . . . 52
3.15 Relative motion geometry, illustrating 1) the relative position between the host and intruder and 2) thee frames of reference: navigation, host and body-fixed frames. . . . . 53
3.16 Sensor suite's field of regard. . . . . 55
3.17 The finite state machine used to model the status of a target track. . . 57
3.18 State machine for the collision avoidance logic, constructed according to Fasano *et al.* [2008]. . . . . 60
3.19 Illustration of conflict detection solution, adapted from Carbone *et al.* [2006]. . . . . 61
3.20 Geometry of a conflict resolution solution, adapted from Luongo *et al.* [2009]. . . . . 64
3.21 The conflict picture of an example encounter. . . . . 65

4.1 The evaluation and analysis framework used throughout this thesis. . . 71
4.2 Notional illustration of all the possible outcomes of an encounter scenario. Solid lines represent the *avoidance trajectories* with the Collision Avoidance System (CAS), while dotted lines represent the *original trajectories* without the CAS, modified from Kochenderfer *et al.* [2010a, Figure 17]. . . . . 72
4.3 Illustrations of the basic concepts in conflict resolution. . . . . 76
4.4 The System Operating Characteristic curves of the candidate collision avoidance logic under four testing conditions. This illustrates the trade-off between the risk ratio and unnecessary alert rate and shows the effect of the limited FOR and of the noise on the logic performance. 80
4.5 Evaluation of the baseline configuration's robustness to noise level. . 82

4.6 Composition of resulting Near Mid-Air Collision (NMAC) events, illustrating the FOR effect on safety performance. . . . . 83
4.7 The resulting expected feedback quality metrics, illustrating the FOR effect on feedback quality. . . . . 83
4.8 The resulting metrics for the probability of Clear of Conflict (CoC) and Secondary Conflict, illustrating the FOR effect on operational suitability. 84

5.1 Three possible models for the protected volume. . . . . 91
5.2 Construction of the generating ellipse of the oblate spheroid model of the collision volume. . . . . 92
5.3 An empirical approach to propagate the intruder uncertainty. . . . . 93
5.4 Illustration of the Converging and Invisibility Cost Functions.. . . . . 97
5.5 Illustration of the final position's coordinate transformation of and limit imposition. . . . . 107
5.6 Squared 2-norm penalty function contour plot with a two-dimensional constraint vector. . . . . 111
5.7 Average cost values of two test runs and their relative optimality. . . . 115
5.8 Histogram of the computational load required by the trajectory planning algorithm with a budget of 4500 function evaluation. . . . . 116
5.9 Maximum constraint violations summary over all infeasible trajectory results. . . . . 117
5.10 Statistics of the initial infeasibility. . . . . 118
5.11 An example trajectory showing the recovery from an initial infeasibility. Although the initial throttle of the reduced-order model is $-0.283$, the subsequent throttles fall back into the feasible region as soon as the second time step. . . . . 118

6.1 Architecture of two conflict resolution functions. . . . . 122
6.2 The system description of the Trajectory Planner. . . . . 123
6.3 Result histograms of the (a) computation time, (b) number of function evaluations and (c) cost function values of three different implementations of the Trajectory Planning Algorithm (TPA): (1) TPA in MATLAB and (2) TPA in MATLAB executable (MEX). . . . . 123

6.4 Trajectory Manager architecture, consisting of Plan Generator, Plan Assessor and Plan Selector. . . . . 124
6.5 The Trajectory Manager's concept of operation. . . . . 124
6.6 Notation used for trajectory tracking, see the first paragraph in §6.2.3 for more details. . . . . 125
6.7 2-Degree of Freedom (DoF) control architecture for the Trajectory Tracker. . . . . 125
6.8 Architecture of the PID controller with saturation and anti-windup. . . 127
6.9 Comparision of two resulting trajectories with and without the feedback compensator, validating the capability of the Trajectory Tracker. 127
6.10 Deliberative and reactive logics' sensitivity curves on Risk Ratio while varying the feedback error level. . . . . 132
6.11 NMAC events decomposition for the three candidate logics. . . . . 134
6.12 Sensitivity curves of probability of resolution-related NMAC, while varying the noise level. . . . . 134
6.13 Sensitivity curves of probability of FOR-related NMAC, while varying the noise level. . . . . 135
6.19 Example encounter illustrating a failure case when using the reactive logic's emergency strategy. Resolution 1 is the reactive logic in cooperative cases. Resolution 2 is the deliberative logic in cooperative cases. . . . . 136
6.14 Sensitivity curves of expected feedback availability level, while varying the noise level. . . . . 137
6.15 Sensitivity curves of expected feedback error level, while varying the noise level. . . . . 137
6.16 Sensitivity curves of probability of clear of conflict, while varying the noise level. . . . . 137
6.17 Sensitivity curves of probability of secondary conflict, while varying the noise level. . . . . 137
6.18 Details of the example encounter in Figure 6.19, while using the reactive logic's emergency strategy. . . . . 137
6.20 Comparison of two uncertainty propagation approaches. . . . . 138

6.21 Example encounter illustrating the conservativeness of the uncertainty propagation approach used. Resolution 1 is the deliberative logic and Resolution 2 is the reactive. . . . . 138
6.22 An observed causal chain from the near-parallel geometry to the resulting NMAC encounters. . . . . 139
6.23 Example encounter illustrating the **parallel-track** result. Resolution 1 is the reactive logic in non-cooperative cases. Resolution 2 is the deliberative logic in non-cooperative cases. . . . . 139
6.24 An observed causal chain illustrating the effect of FOR restriction on the feedback quality and safety performance. . . . . 140
6.25 Example encounter illustrating the **FOR effect**: 1) the effect of manoeuvring on feedback availability; and 2) the effect of feedback availability on feedback error. Resolution 1 is the reactive logic (at 2Hz) and Resolution 2 is the deliberative logic. Both are in non-cooperative cases with noise at level 5. . . . . 140

B.1 Flight envelop of the Jetstream 31 aircraft model used in this work. . . 163
B.2 The maximum load factor look-up table model. . . . . 163
B.3 The n-V diagram used as the maximum load factor model used in the trajectory planning algorithm. . . . . 164
B.4 The step responses of three control channels: Airspeed, Aircraft Heading, and Aerodynamic Flight-Path Angle. . . . . 165
B.5 The ramp responses of the three command channels. . . . . 167
B.6 The responses for a three dimensional manoeuvre with simultaneous changes in three control channels. . . . . 168

C.1 A feasible trajectory example. . . . . 175
C.2 An acceptable trajectory example. Most performance violations are within 2% of their limits; and at its maximum constraint violation, the miss distance is 591 ft, that is 91 ft further than the required separation. 175
C.3 An infeasible-obstacle trajectory example. Despite the 49% of obstacle constraint violation, the estimated miss distance and the required performance show the trajectory's applicability. . . . . 175
C.4 An infeasible-performance trajectory example. . . . . 175

C.5 A both-infeasible trajectory example. . . . . 176

This page intentionally contains only this sentence.

# List of Tables


2.1 Typical SAA sensor characteristics [Chen *et al.*, 2011]. . . . . 19

3.1 Part of the model variables specifying the encounter characteristics. . 35
3.2 Conditional probability table of a turn and speed change. . . . . 36
3.3 Intruder limitations parameters. . . . . 39
3.4 An example set of surveillance sensor parameters. . . . . 55
3.5 Measurement error model parameters. . . . . 58
3.6 Guarding conditions for the transitions among different finite states of the baseline collision avoidance logic. . . . . 66
3.7 Summary sheet of the models used for Monte-Carlo simulation (I). . . 68
3.8 Summary sheet of the models used for Monte-Carlo simulation (II). . 69

4.1 Event definitions for the possible outcomes of results. . . . . 73
4.2 Characteristics of the original aircraft trajectories in the standard encounter set. . . . . 78
4.3 Test condition definitions. . . . . 78
4.4 Summary of twelve configurations for the candidate collision avoidance logic. . . . . 79
4.5 Logic performance of the baseline configuration, under the nominal noise level. . . . . 81

5.1 Success rate for the testing of the trajectory planning algorithm over 1040 test cases. . . . . 114
5.2 Summary of the average computational load for the trajectory planning algorithm over 1024 testing scenarios. . . . . 116

5.3 Percentage of five types of result trajectories, categorized according to their constraint violations. . . . . 117

6.1 The parameter values for the PID controllers in the Trajectory Tracker. 127
6.2 Experiment settings for the safety performance evaluation. . . . . 129
6.3 Overall safety performance results of three candidate logics, compared with the original performance with no conflict resolution logic. . . . . 129
6.4 Safety performance in distance-emergency situations. . . . . 130
6.5 Safety performance in time-emergency situations. . . . . 131
6.6 Experiment settings for the robustness evaluation. . . . . 131
6.7 Experiment settings for the evaluation of FOR effect. . . . . 133

7.1 Summary of the objectives, findings and conclusions of the five simulation experiments in this thesis. . . . . 151
7.2 The main result of the initial feasibility study of the proposed approach—based on a trajectory-planning method for robust collision avoidance with high performance manoeuvres, as described in §1.3. . . . . 152

B.1 Aircraft performance model parameters. . . . . 164

C.1 Trajectory Planning Algorithm Parameters. . . . . 174

This page intentionally contains only this sentence.

# Nomenclature

**Typeface**

| | |
|---|---|
| $\mathsf{A}$ | A coordinate system. |
| $\mathcal{A}$ | A frame. |
| $\mathtt{A}$ | A point. |
| $\underline{\mathbf{A}}$ | A matrix. |
| $\vec{\boldsymbol{a}}$ | A Euclidean vector. |
| $\boldsymbol{a}$ $\boldsymbol{\alpha}$ | A vector. |
| $a$ $\alpha$ | A scalar. |

**Accent**

| | |
|---|---|
| $\dot{(\cdot)}$ | A derivative with respect to time |
| $\hat{(\cdot)}$ | An estimation |
| $\tilde{(\cdot)}$ | A measurement or variable subject to random error |

**Coordinate System**

| | |
|---|---|
| $\mathsf{B_s}$ | Body-fixed spherical coordinate system |
| $\mathsf{B}$ | Body-fixed coordinate system |
| $\mathsf{C}$ | Conflict coordinate system |
| $\mathsf{D}$ | Desired-path coordinate system |
| $\mathsf{H}$ | Host-carried coordinate system |
| $\mathsf{N}$ | Navigation coordinate system |
| $\mathsf{R}$ | Resolution coordinate system |

**Euclidean vector - Roman letters**

| | |
|---|---|
| $\vec{\boldsymbol{r}}_{\mathtt{A/B}}$ | The displacement vector of a point $\mathtt{A}$ with respect to a point $\mathtt{B}$ |
| $\vec{\boldsymbol{p}}_{\mathtt{A}/\mathcal{H}}$ | The position vector of a point $\mathtt{A}$ in a frame $\mathcal{H}$ |

**Frame - Roman letters**

| | |
|---|---|
| $\mathcal{B}$ | Body-fixed frame. |
| $\mathcal{H}$ | Host-carried frame. |
| $\mathcal{N}$ | Navigation (inertial) frame. |

**Matrix - Roman letters**

| | |
|---|---|
| $\underline{\mathbf{C}}_{\mathsf{A}}^{\mathsf{B}}$ | The direction cosine matrix from a coordinate system $\mathsf{A}$ to a coordinate system $\mathsf{B}$ |

**Point - Roman letters**

| | |
|---|---|
| `B` | The origin of the body-fixed coordinate system |
| `D` | Current desired position point |
| `H` | The origin of the host-carried coordinate system; the host aircraft's center of mass |
| `I` | The center of mass of the intruder aircraft |
| `N` | The origin of the navigation coordinate system |
| `R` | The origin of the resolution coordinate system |

**Right subscripts**

| | |
|---|---|
| $(\cdot)_0$ | Of the initial state |
| $(\cdot)_D$ | Of the Down coordinate |
| $(\cdot)_E$ | Of the East coordinate |
| $(\cdot)_N$ | Of the North coordinate |
| $(\cdot)_{\mathtt{H}}$ | Of the host |
| $(\cdot)_{\mathtt{I}}$ | Of the intruder |
| $(\cdot)_f$ | Of the final state |
| $(\cdot)_{rou}$ | Of the original route |
| $(\cdot)_{max}$ | Of the maximum limitation |
| $(\cdot)_{min}$ | Of the minimum limitation |

**Right superscripts**

| | |
|---|---|
| $(\cdot)^{CR}$ | For conflict resolution |
| $(\cdot)^{des}$ | For the desired values |
| $(\cdot)^{ref}$ | For the reference values |

**Scalar - Greek letters**

| | |
|---|---|
| $\Phi_{B_s}$ | Intruder's azimuth coordinate in the body-fixed spherical coordinate system |
| $\Phi_{lim}$ | Azimuth limitation of the field of regard of a surveillance sensor |
| $\Theta_{B_s}$ | Intruder's elevation coordinate in the body-fixed spherical coordinate system |
| $\Theta_{lim}$ | Elevation limitation of the field of regard of a surveillance sensor |
| $\alpha$ | Angle of attack |
| $\chi$ | Ground track angle |
| $\epsilon$ | Random error |
| $\lambda_{ava}$ | Feedback availability |
| $\lambda_{ava}$ | Feedback error |
| $\mu$ | Bank angle |
| $\phi_f$ | Final cost |
| $\phi$ | Roll attitude |
| $\psi_f$ | Final constraint |
| $\psi$ | Yaw attitude |
| $\rho$ | Air density; penalty parameter |
| $\sigma$ | Standard deviation |
| $\tau$ | Time constant of the of the uncertainty bound of the intruder projection model |
| $\theta$ | Pitch attitude |

**Scalar - Roman letters**

| | |
|---|---|
| $CD$ | Number of Correct Detection events |
| $CR$ | Number of Correct Rejection events |
| $C_{D0}$ | Zero-lift drag coefficient |
| $C_D$ | Drag coefficient |
| $C_{L0}$ | Zero-angle-of-attack lift coefficient |
| $C_{L\alpha}$ | Lift curve slope |
| $C_L$ | Lift coefficient |
| $C_{T_{c,i}}$ | Thrust coefficients |
| $C_{T_{cr}}$ | Thrust coefficient |
| $D$ | Drag force |
| $FA$ | Number of False Alarm events |

| | |
|---|---|
| $F$ | Cost of the Nonlinear Programming problems |
| $H$ | Required vertical separation |
| $IN$ | Number of Induced Near Mid-Air Collision events |
| $J$ | Cost functional |
| $K$ | Controller gain |
| $LA$ | Number of Late Alert event. |
| $L$ | Lift force |
| $MD$ | Number of Missed Detection events |
| $M$ | Mach number |
| $NM$ | Number of Near Mid-Air Collision even. |
| $RR$ | Risk ratio |
| $R$ | Required horizontal separation |
| $S$ | Aircraft wing area |
| $T$ | Thrust, propulsion force |
| $UL$ | Uncertainty Level |
| $V_a$ | True airspeed |
| $V_g$ | Ground speed |
| $V_v$ | Vertical speed |
| $V$ | Inertial speed |
| $\mathrm{E}(\lambda_{ava})$ | Expected feedback availability |
| $\mathrm{E}(\lambda_{err})$ | Expected feedback error |
| $\bar{T}$ | Throttle setting |
| $\chi_a$ | Aerodynamic heading angle |
| $\dot{r}$ | Relative range rate |
| $\gamma_a$ | Aerodynamic flight-path angle |
| $\gamma$ | Flight-path angle |
| Pr(CoC) | Probability of Clear of Conflict, a.k.a. CoC rate |
| Pr(NM) | Probability of Near Mid-Air Collision, a.k.a. NMAC rate |
| Pr(SC) | Probability of Secondary Conflict, a.k.a. secondary conflict rate |
| Pr(UA) | Probability of Unnecessary Alert, a.k.a. unnecessary alert rate |
| $a$ | Length of the major axis of the generating ellipse of the spheroid protected volume |

| | |
|---|---|
| $b$ | Length of the minor axis of the generating ellipse of the spheroid protected volume |
| $d_{\mathsf{C}_{lin}}$ | Linear miss distance |
| $g$ | Acceleration due to gravity, or the growth factor of the uncertainty bound of the intruder projection model |
| $hmd$ | Horizontal miss distance |
| $h$ | Altitude |
| $k$ | Induced drag factor |
| $m$ | Mass |
| $n_z$ | Normal load factor |
| $p_1$ | Host's first coordinate in the resolution coordinate system |
| $p_2$ | Host's second coordinate in the resolution coordinate system |
| $p_3$ | Host's third coordinate in the resolution coordinate system |
| $r_{\mathsf{B}_\mathsf{s}}$ | Intruder's radial coordinate in the body-fixed spherical coordinate system |
| $r_{max}$ | Relative range limitation |
| $r$ | Radial coordinate in a spherical coordinate system; or relative range between the host and intruder |
| $t_{\mathsf{C}_{lin}}$ | Linear time-to-CPA |
| $tca$ | Time of closest approach |
| $t$ | Time |
| $vmd$ | Vertical miss distance |

**Vector - Greek letters**

| | |
|---|---|
| $\Psi$ | Aircraft attitude vector |
| $\boldsymbol{\Xi}$ | Optimization variable vector |
| $\boldsymbol{\epsilon}$ | Random error vector |
| $\boldsymbol{\psi}_c$ | Trajectory control constraint vector |
| $\boldsymbol{\psi}_d$ | Trajectory derivative constraint vector |
| $\boldsymbol{\psi}_o$ | Trajectory obstacle constraint vector |
| $\boldsymbol{\psi}_x$ | Trajectory state constraint vector |
| $\boldsymbol{\psi}$ | Trajectory constraint vector |

**Vector - Roman letters**

| | |
|---|---|
| $\boldsymbol{c}^+$ | Constraint violation vector |

| | |
|---|---|
| $\boldsymbol{c}_{\mathtt{H}}$ | Host manoeuvre command vector |
| $\boldsymbol{c}_{\mathtt{I}}$ | Intruder manoeuvre command vector |
| $\boldsymbol{c}$ | Constraint vector |
| $\boldsymbol{e}$ | Error vector |
| $\boldsymbol{p}_{\mathtt{I/B_s}}$ | The relative position vector in the body-fixed spherical coordinate system |
| $\boldsymbol{p}_{\mathtt{I/B}}$ | The relative position vector in the body-fixed coordinate system |
| $\boldsymbol{p}_{\mathtt{I/H}}$ | The relative position vector in the host-carried coordinate system |
| $\boldsymbol{p}_{\mathtt{I}}$ | Intruder position vector in the navigation (or resolution) coordinate system |
| $\boldsymbol{p}_r$ | Relative position vector of the intruder with respect to the host in navigation (or resolution) coordinate system |
| $\boldsymbol{v}_{\mathtt{H}}$ | Host velocity vector in the navigation coordinate system |
| $\boldsymbol{v}_{\mathtt{I}}$ | Intruder velocity vector in the navigation coordinate system |
| $\boldsymbol{v}_r$ | Relative velocity vector of the intruder with respect to the host in navigation (or resolution) coordinate system |
| $\boldsymbol{x}_{\mathtt{H}}$ | host state vector |
| $\boldsymbol{x}_{\mathtt{I}}$ | intruder state vector |
| $\boldsymbol{x}_{nav}$ | Navigation state vector |
| $\boldsymbol{x}_{tra}$ | Tracker state vector |
| $\boldsymbol{z}_{\mathtt{I}}^{lim}$ | Intruder limited-field-of-regard measurement vector |
| $\boldsymbol{z}_{\mathtt{I}}$ | Intruder measurement vector |

This page intentionally contains only this sentence.

# Acronyms

| | |
|---|---|
| **ACAS** | Airborne Collision Avoidance System |
| **ACAS X** | Airborne Collision Avoidance System X |
| **ADS-B** | Automatic Dependent Surveillance-Broadcast |
| **ASAS** | Airborne Separation Assurance System |
| **ATC** | Air Traffic Control |
| **ATCO** | Air Traffic Controller |
| **ATM** | Air Traffic Management |
| **BADA** | Base of Aircraft Data |
| **CA** | Collision Avoidance |
| **CAA** | Civil Aviation Authority |
| **CAS** | Collision Avoidance System |
| **CASSATT** | Collision Avoidance System Safety Assessment Tool |
| **CDR** | conflict detection and resolution |
| **CM** | Centre of Mass |
| **CoC** | Clear of Conflict |
| **CPA** | Closest Point of Approach |
| **DoF** | Degree of Freedom |
| **EO** | Electro-Optical |
| **EoM** | Equations of Motion |
| **EUROCONTROL** | European Organisation for the Safety of Air Navigation |
| **FAA** | Federal Aviation Administration |

| | |
|---|---|
| **FOR** | Field of Regard |
| **FoV** | Field of View |
| **FPA** | Flight Path Angle |
| **HJ** | Hooke-Jeeves |
| **HMD** | Horizontal Miss Distance |
| **HMI** | Human-Machine Interface |
| **ICAO** | International Civil Aviation Organization |
| **IR** | Infra-Red |
| **ISA** | International Standard Atmosphere |
| **JOCA** | Jointly Optimal Collision Avoidance |
| **MAC** | Mid-Air Collision |
| **MEX** | MATLAB Executable |
| **MIDCAS** | Mid Air Collision Avoidance System |
| **NDI** | Nonlinear Dynamic Inversion |
| **NLP** | Nonlinear Programming |
| **NMAC** | Near Mid-Air Collision |
| **OCP** | Optimal Control Problem |
| **OV** | Optimization Variable |
| **SAA** | Sense and Avoid |
| **SAAFT** | Sense-and-Avoid Flight Tests |
| **SOC** | System Operating Characteristic |
| **SS** | Self Separation |
| **TCAS** | Traffic Alert and Collision Avoidance Systems |
| **TCAS II** | Traffic Alert and Collision Avoidance Systems II |
| **TGA** | Trajectory Generation Algorithm |
| **TGP** | Trajectory Generation Problem |
| **TPA** | Trajectory Planning Algorithm |
| **TPP** | Trajectory Planning Problem |
| **TRL** | Technology Readiness Level |

| | |
|---|---|
| **UAS** | Unmanned Aircraft Systems |
| **UAV** | Unmanned Aerial Vehicle |
| **UAVp** | Unmanned Aerial Vehicles pilot |
| **VMD** | Vertical Miss Distance |

This page intentionally contains only this sentence.

# Chapter 1

# Introduction

## 1.1 Motivations

To fully realize their great potential for dirty, dull and dangerous tasks, Unmanned Aircraft Systems (UAS) require their routine access to all classes of airspace without the restrictive conditions of operation. According to the existing regulations and guidance by different organizations [FAA, 2008; Drozdowski & Dean, 2010; CAA, 2012], UAS are required to comply with the same flight rules as manned aircraft. This includes a requirement to 'see and avoid' other airspace users as specified in the Rules of the Air [ICAO, 2005a]. Currently, in order to satisfy this requirement, a UAS can only operate in the segregated airspace, where collision risks are eliminated by strictly controlling entry to this airspace by other aircraft; otherwise, it is required to be equipped with an approved Sense and Avoid (SAA) systems as a mean to comply with the see and avoid requirement. Despite tremendous efforts that have been devoted to the integration of UAS in non-segregated airspaces, the standard for SAA systems has yet to be established and the SAA remains one of the unconquered challenges. This challenge is mentioned in the recent UAS integration roadmap by [Huerta, 2013]:

> *Research is underway on Airborne Sense and Avoid (ABSAA) concepts. Due to complexity, significant progress in ABSAA is not expected until the mid-term [between 2015-2020]. Research goals for the near-term [prior to October 2015] include a flight demonstration of various sensor modes (electro-optic/infrared, radar, Traffic Alert and Collision Avoidance*

> *System (TCAS) and Automatic Dependent Surveillance-Broadcast (ADS-B)). Actual fielding of a standardized ABSAA system is a long-term [after 2020] objective.*

The sensing function of the SAA capability involves monitoring the surrounding traffic and maintaining a track for every detected object. Similar function is referred to as the Surveillance and Tracking Module in the Airborne Collision Avoidance System X (ACAS X), the next generation airborne collision avoidance systems [Walters, 2012]. Although still some time away from standardization, the development of the SAA sensing function is relatively mature and appears to settle on a *multi-sensor, data-fusion configuration* with the sensor technologies mentioned in the previous paragraph. This multi-sensor configuration is adapted by all the reviewed state-of-the-art SAA technology demonstration programmes, such as [Chen *et al.*, 2011; Buchanan, 2012; Petri & Spriesterbach, 2012; MIDCAS, 2013].

On the other hand, the avoidance function of SAA capability is relatively in its infancy. A key element of the avoidance function is the *collision avoidance logic*, which relies on noisy estimations from the sensing function to determine whether a collision alert needs to be issued and, if so, what resolution action shall be taken. It is said to be in its infancy not because of the lack of collision avoidance logic solutions—in fact numerous collision avoidance logics in the literature are shown to be effective—but because of the challenge to accommodate the unique characteristics of UAS, for instance, the absence of an onboard pilot, the diversity in unmanned aircraft's dynamics performance and in sensor capabilities, and the issues of communication latency and possible lost links [Zeitlin, 2012; Edwards, 2012].

In order to mitigate the impact of delayed or lost link issues, SAA systems are expected to be required to perform an automatic collision avoidance manoeuvre after reaching some design threshold, see NATO [2008, CAS9] and Sellem-Delmar [2010, §12.7.2]. However, automatic collision avoidance manoeuvres would only be allowed if the false alarm rate is acceptably low. For instance, in the design of an automatic aircraft collision avoidance logic for fighter aircraft [Sundqvist, 2005; Turner *et al.*, 2012], a nuisance free requirement is posed so that the automatic manoeuvre would not interfere the current operational mission.

Furthermore, the false alarm rate is traded directly with the safety performance, i.e. for a particular collision avoidance logic design, adjusting the parameters to allow a

smaller false alarm rate would inevitably increase the collision risk. In other words, the envisioned, more stringent requirement on false alarm would make a considerable number of existing collision avoidance logics unsuitable candidates for SAA systems.

Therefore, the motivation of this thesis is to develop a collision avoidance logic that is able to not only prevent collision but also achieve the envisioned, stringent false alarm rate for automatic activation.

## 1.2 Challenges

In the development of new collision avoidance logics, there is a trend to use a trajectory planning framework, which is characterized by the following five elements: 1) a response model of the host aircraft to generate the avoidance trajectory; 2) a dynamic model of the intruder to predict its future state; 3) some criteria used to evaluate the feasibility and optimality of the avoidance trajectory; 4) an action space to define all the available avoidance manoeuvres; and 5) an online mechanism to determine the optimal action to perform. Despite the slight differences in their realizations, the solutions proposed by different SAA technology demonstration programmes [Chen *et al.*, 2009; Nicoullaud, 2012; Petri & Spriesterbach, 2012] also fit into this framework.

Although the trajectory planning framework normally relies on some computationally expensive methods, there is a desire to increase the update rate so that the effect of the uncertainty in the intruder state can be alleviated. In order to satisfy a target update rate, most trajectory planning methods in the literature have to restrict their solution spaces to the ones with only a small number of resolution actions, and thus sacrificing the aircraft manoeuvrability that is safely available for collision avoidance.

On the other hand, it is believed that making the most of the aircraft performance that is safely available for collision avoidance can improve the safety performance and also alleviate the difficulty in meeting the stringent nuisance alert rate requirement. For instance, MIDCAS [Nicoullaud, 2012, pp. 23] proposes to perform the automatic collision avoidance manoeuvre at the last instant associated with a high performance manoeuvre, so that the nuisance alert rate can be reduced and it leaves the maximum amount of time for UAS pilots to use their normal authorities. Figure 1.1a shows a notional example to illustrate how high performance manoeuvres can safely delay an automatic collision avoidance manoeuvre. Figure 1.1b illustrates how the problem

dimension is increasing with the fidelity of the host aircraft model.

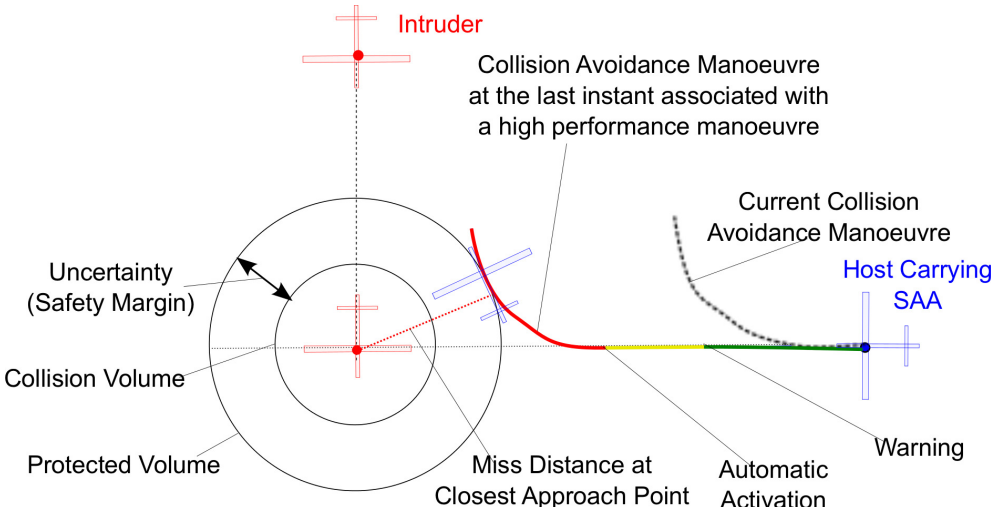
However, subject to the 'curse of dimensionality', it is daunting to extend the existing trajectory planning methods to handle the collision avoidance problems of high dimension in real time. These challenges have also been mentioned by Temizer [2011]; Wolf & Kochenderfer [2011]; Bai & Hsu [2011]; Chen *et al.* [2009].

## 1.3 Overview of the Proposed Approach

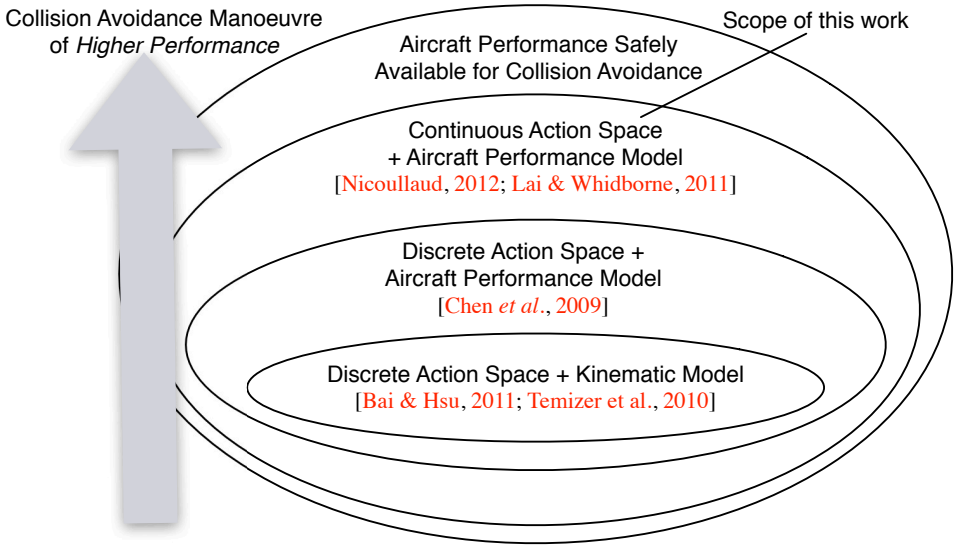
In order to handle the high dimensional collision avoidance problems in real time, this thesis proposes to use a control architecture to separate the avoidance trajectory planning from the avoidance trajectory execution and to use a trajectory planning method, based on the real-time approximate solution for an optimal control problem, to take into account the flyability of high performance manoeuvres and robustness to the intruder projection uncertainty.

Firstly, the control architecture, as shown in Figure 1.2, is based on the conventional robotic *three-layer architecture* (see Gat [1998]; Patchett & Ansell [2010]), comprising Planning, Management (or Sequencing) and Control. Starting from the bottom is the control layer containing a typical navigation, guidance and control loop, which is used to automate the execution of the avoidance command from the decision maker and to assure an accurate execution of the high performance avoidance manoeuvre, see §6.2.3 for more details. The management layer contains a decision making loop, which is in charge of making the decision about when the avoidance action should be commanded, see Figure 2.7 and §6.2.2 for more details. The planning layer accommodates the proposed trajectory planning method, which is responsible for answering the request from the management layer with an avoidance trajectory.

Secondly, there are three main steps in the robust trajectory planning method, as shown in Figure 1.2. The method starts with generating a candidate trajectory with an initial guess of the avoidance action. The candidate trajectory is then evaluated against a set of criteria representing the constraints and costs defined for the collision avoidance problem, some example criteria are shown in Figure 1.3 and more details can be found in §5.2. Based on the evaluated values of the criteria, an optimization mechanism is employed to iteratively adjust the avoidance action, until it finds an avoidance trajectory whose constraint and cost values satisfy some specified conditions.




(a) A notional example illustrating how a high performance manoeuvre can safely delay an automatic collision avoidance manoeuvre, adapted from Nicoullaud [2012].




(b) The notional solution spaces of different collision avoidance logics in the literature, and their relationship to high performance avoidance manoeuvres.

Figure 1.1: The notional explanation of the desire to use an avoidance manoeuvre of higher performance.

The trajectory planning method is robust in the sense that it uses an empirical method, i.e. linear projection with an exponentially growing uncertainty margin, to project the intruder state so as to explicitly compensate for prediction uncertainty.

The main concepts and key features of the robust trajectory planning method are summarized in Figures 1.3 and 1.4, the details of which can be found in §5.2. The detailed implementation of the proposed control architecture can be found in §6.2.

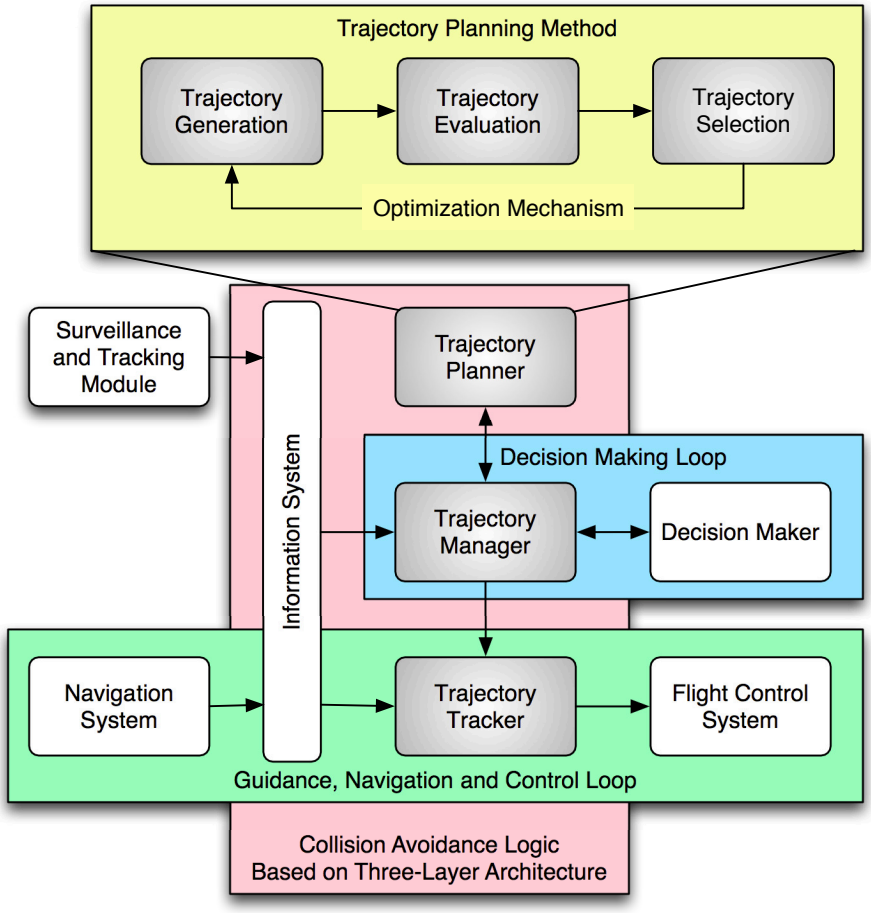



Figure 1.2: Architecture of the proposed collision avoidance logics for SAAs systems. The grey functional blocks highlight the focus of this thesis. More details can be found in §2.3.3 and §6.2 for the architecture and in Chapter 5 for the trajectory planning method.

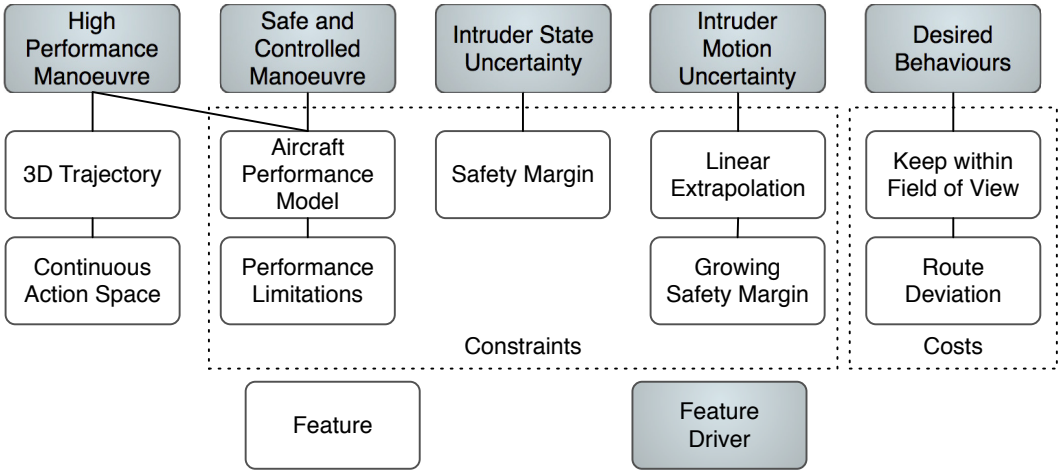



Figure 1.3: Main features of the proposed trajectory planning method, more details can be found in Chapter 5.

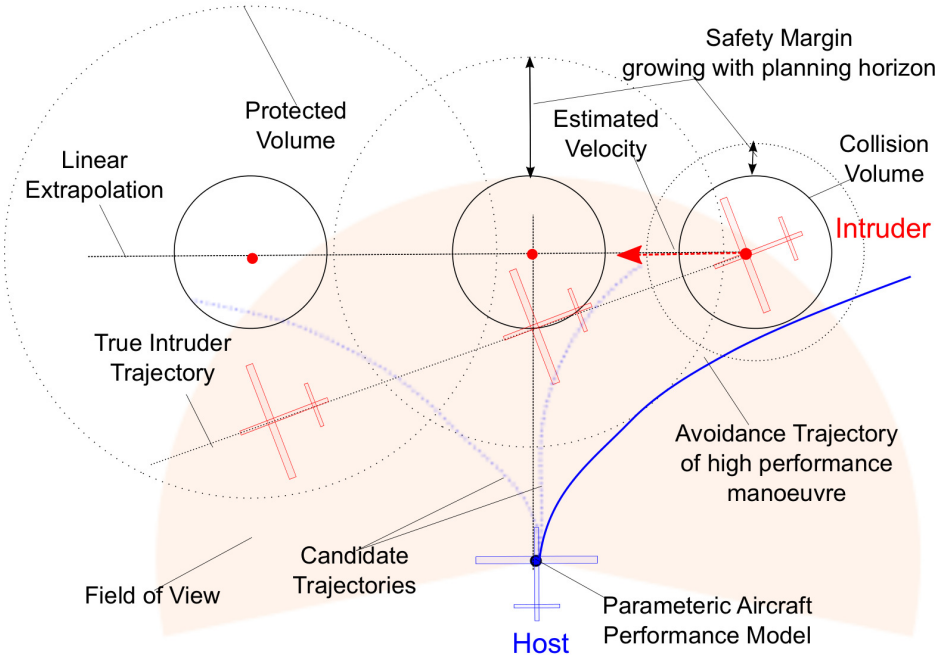



Figure 1.4: Concept of the proposed trajectory planning method of high performance manoeuvres for collision avoidance problems.

## 1.4 Aim and Objectives

The aim of this thesis is to develop a real-time trajectory planning algorithm for a novel Unmanned Aerial Vehicle (UAV) collision avoidance logic and to determine the integrated logic's feasibility, merits and limitations for practical applications. This was achieved by pursuing the following objectives:

1. Perform a review of the latest collision avoidance logics and the existing methods for trajectory planning methods so as to determine the problem scope and identify the potential methods;

2. Construct a modelling and simulation framework[1] for the performance evaluation of collision avoidance logics; the framework should be sufficiently representative of the realistic operating environment, and yet computationally fast and flexible enough for large-scale, fast-time Monte Carlo simulations;

3. Establish a set of effectiveness measures and a baseline for performance comparison; the effectiveness measures should be recognizable in the research community of collision avoidance systems and the baseline method should be comparable with the proposed method;

4. Develop a trajectory planning algorithm that is able to take into account the flyability of combined manoeuvres, robustness to intruder uncertainty and Field of Regard (FOR) restriction; the selected algorithm should have the potential for real-time implementation; and

5. Evaluate the performance of the integrated system so as to determine the feasibility, merits and drawbacks of the proposed method.

## 1.5 Organization and Highlights of this Thesis

Figure 1.5 provides an overview of this thesis and the highlights of each chapter is summarized as follows:

[1]By framework, this thesis means a collection of computational models and analysis tools designed for a particular purpose.

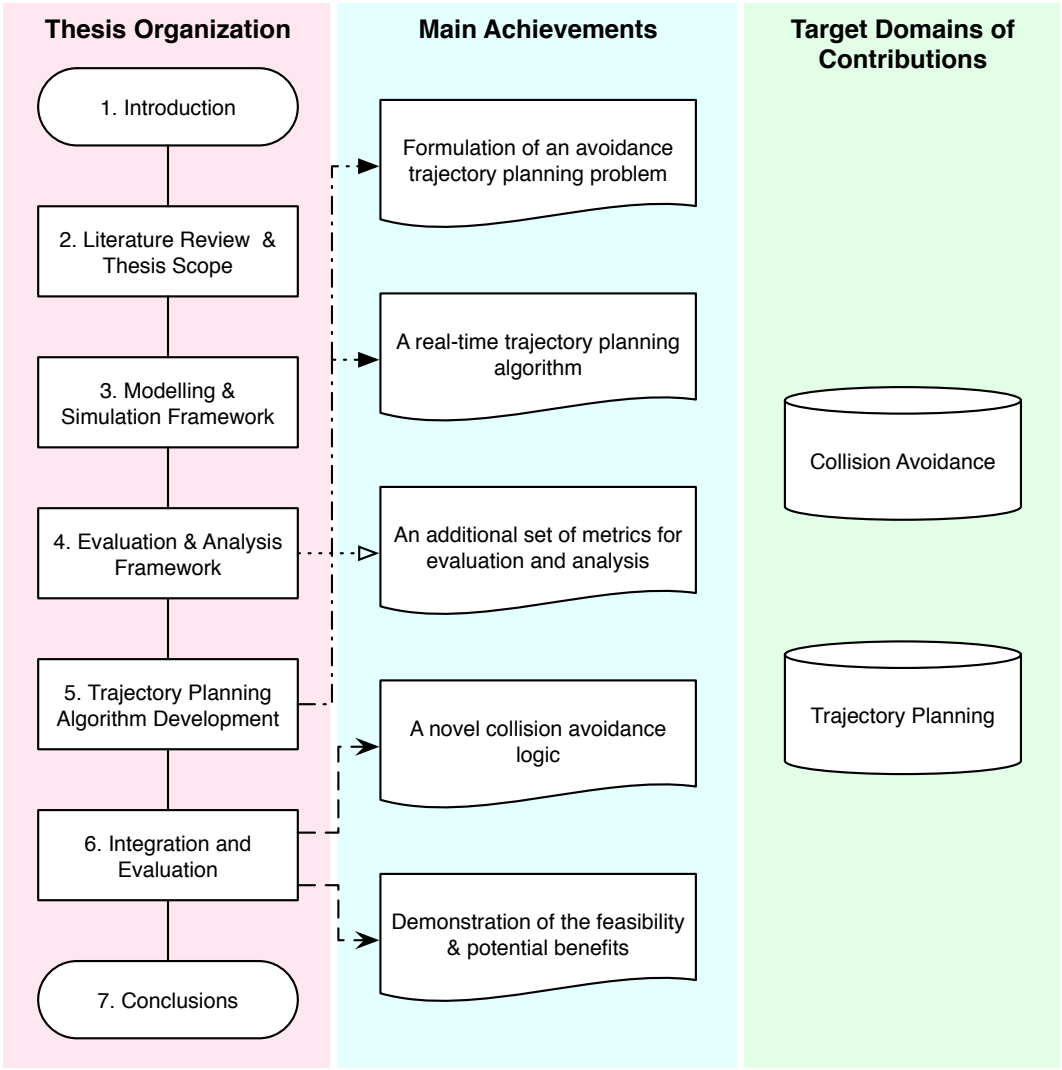



Figure 1.5: Overview of the organization, deliverables and target domains of contributions of this thesis, see §7.2 for more details.

### Chapter 2: Literature Review and Thesis Scope

In order to confine the scope of this work, this chapter first presents the background information about the role of collision avoidance in the current Air Traffic Management (ATM) system (§2.2); and then summarizes the development process, intended functions and functional architecture of SAA systems in §2.3. Moreover, it reviews the latest work published in the field of collision avoidance logic development in §2.3 and describes the reason for the choice of the trajectory planning algorithm.

### Chapter 3: Modelling and Simulation Framework

For the capability to statistically demonstrate the effectiveness of the safety-critical collision avoidance systems, this chapter aims to construct a modelling and simulation framework for large-scale, fast-time Monte Carlo simulations. In order to emulate the actual operating conditions, the following models are implemented: 1) the Standard Encounter Model used by ICAO [2007] to generate a credible set of testing scenarios (§3.2); 2) a kinematic intruder model with configurable aircraft performance and manoeuvre uncertainty (§3.3); 3) an aircraft system model with configurable transient response characteristic and realistic aircraft performance according to the Base of Aircraft Data (BADA) (§3.4); 4) a surveillance and tracking model able to produce the noisy and limited FOR[1] surveillance conditions (§3.5); and 5) a collision avoidance logic model implementing a *collision-cone-like*[2] geometric conflict detection and resolution method §3.6.

The final result of this chapter is a modular and parametric framework. As summarized in Tables 3.7 and 3.8, the implementation used in this thesis is based on the ICAO [2007] Standard Encounter Model and the aircraft performance model of the Jetstream 31 aircraft [Nuic, 2012; Cooke, 2008].

[1]Field of regard is the spatial volume, within which the SAA sensor(s) can make measurements, which is specified in terms of relative range, azimuth and elevation from the fixed body reference frame of the SAA platform, see §3.5.2 for specific details.

[2]See Carbone *et al.* [2006]; Chakravarthy & Ghose [1998]

**Chapter 4: Evaluation and Analysis Framework**

Given the enormous amount of result data from a Monte Carlo simulation, this chapter presents a framework, consisting of a systematic process (Figure 4.1) and a collection of analysis tools, to facilitate the performance evaluation and safety analysis of collision avoidance logics. A typical set of performance metrics, such as *risk ratio* and *unnecessary alert rate*, is first presented in §4.2; and then, in order to enable the investigation of the effect of the FOR restriction on the logic performance, an additional set of metrics (such as the probability of a secondary conflict and feedback quality) is introduced in §4.3. Finally, other necessary analysis tools (such as System Operating Characteristic (SOC) and sensitivity curves) along with two simulation experiments—establishment of baseline performance and investigation of FOR-restriction effect[1]—are described in 4.4.

Besides the establishment of the baseline performance for comparison, the main contribution of this chapter is to quantitatively show that the FOR restriction would reduce the quality of the feedback to the collision avoidance logic, increase the risk of collision and make the determination of the moment to issue a Clear of Conflict (CoC) more challenging (4.4.3).

**Chapter 5: Trajectory Planning Algorithm Development**

In order to make the most of the aircraft performance that is safely available for collision avoidance, this chapter describes the development of a trajectory planning algorithm that is capable of planning resolution trajectories with high performance manoeuvres. The development involves 1) capturing some requirements from the literature, such as the flyability of resolution manoeuvres, minimum separation and operational suitability (§5.1), 2) formulating an optimal control problem to accommodate the identified requirements (§5.2), and 3) implementing an algorithm to solve the optimal control problems.

The algorithm itself contains an *inverse-dynamic direct method* (§5.3) to convert the optimal control problem to a 15-dimensional Nonlinear Programming (NLP) problem (Problem 5.3.1) and a two-stage, derivative-free pattern search method §5.4.4 to solve the converted Nonlinear Programming (NLP) problems.

[1]The effect caused by the FOR restriction.

The testing results (§5.5) show that the algorithm (without the low-level code generation) is able to find a feasible trajectory in 88% of the testing encounters, using 2.56 seconds and 2212 function evaluations on average. With the function evaluation budget of 4500, the maximum computation time over all cases is 5 seconds. A preliminary investigation of the usability of the reaming 12% infeasible solutions is also presented.

**Chapter 6: System Integration and Performance Evaluation**

In order to provide the initial feasibility study with evidence, this chapter first describes an integrated software prototype for testing and then presents the testing results of a Monte Carlo simulation with 3800 encounter geometries under 8 different surveillance conditions (resulting in 30400 testing encounters in total). Section 6.2 first describes the architecture of the overall prototype, consisting of a Trajectory Planner (§6.2.1), a Trajectory Manager (§6.2.2) and a Trajectory Tracker (§6.2.3). In particular, the Trajectory Planner was implemented using low-level code generations and the testing results in §6.2.1 shows that the maximum computation time of the MATLAB executable has been reduced by 10 times to about 0.5 seconds.

Section 6.3 presents the results of three simulation experiments for the purpose of 1) safety performance evaluation in ideal conditions, 2) robustness analysis to sensor noise, and 3) investigation of the FOR-restriction effect. The main results shows that the proposed method outperformed the baseline method, in terms of risk ratio by 50%, and was significantly more robust to sensor noise. Table 7.1 summarizes other findings in the three experiment simulations.

Section 6.4 discusses four phenomena—ineffectiveness of the emergency strategy, conservativeness in uncertainty propagation, FOR-restriction effect and parallel-track encounters—observed from the resulting Near Mid-Air Collision (NMAC) encounters. One major finding in this section is an observed causal chain (Figure 6.24) that explains how the FOR restriction can affect the feedback quality and safety performance.

**Chapter 7: Conclusions**

This chapter draws conclusion on the proposed logic's feasibility and potential benefits, outlines the deliverables from this work, highlights the contributions to knowledge and suggests areas for further development.

# Chapter 2

# Background and Literature Review

## 2.1 Introduction

This chapter determines the problem scope of this thesis via 1) providing the background information about the role of collision avoidance in the current ATM systems; 2) presenting the development process of SAA systems; 3) reviewing the latest development on the collision avoidance logics; and 4) explaining the choice of the trajectory planning algorithm for further development. The organization and the scopes of this thesis are summarized in Figure 2.1.

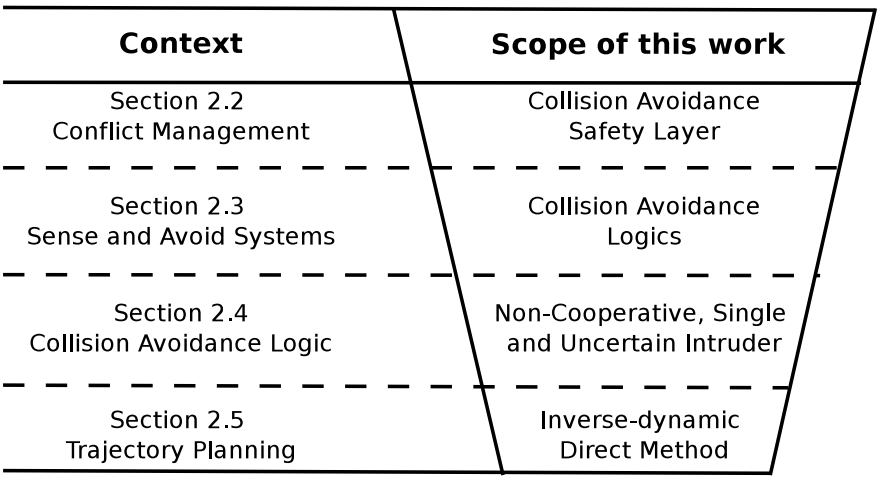



Figure 2.1: Overview of Chapter 2, containing the context of the literature and their relationships to the scope of this thesis.

## 2.2 Conflict Management

### 2.2.1 Layered Approach

In the current ATM system, a number of *risk mitigation measures* (including procedures, personnel and systems) are in place in order to limit the risk of collision to an acceptable level. The mitigation measures are effectively functioning in a *layered approach*, which means that it would take failures at multiple layers to cause a system failure. Figure 2.2 illustrates such a layered approach, where each *safety layer* contains a set of mitigation measures and a hole in the safety layer represents a possible failure of the mitigation measures.

Figure 2.2a shows a *safety layer model* to represents the ATM situation. The model is consisted of three *conflict management* layers as specified by ICAO [2005b] and one final layer of Providence to distinguish an actual mid-air collision from an NMAC, which is defined mainly for analysis purposes.[1]

Figure 2.2b shows a model with an additional layer of *Separation Restoration*, as mentioned by Drozdowski & Dean [2010, p. 26]. This layer is included here to highlight the fact that there may exist a gap in the current model and a set of mitigation measures are emerging to improve the ATM system by filling this gap.

### 2.2.2 Safety Events

For the purpose of safety analyses, the following *safety events* as the consequences of breaching the safety layers are defined:

- **Conflict** is a situation involving aircraft and hazards[2] in which the applicable separation minima **may be** compromised, according to ICAO [2005b].
- **Loss of Separation** is a situation involving aircraft and hazards in which the applicable separation minima **has been** compromised.
- **Close Proximity** is a situation in which the involving aircraft are in such proximity as to require collision avoidance actions to maintain safety, in the opinion

[1]There is about a 1 in 10 chance that an NMAC will in fact be an actual collision, see Drozdowski & Dean [2010, pp.12].

[2]Hazards include other aircraft, terrain, weather, wake turbulence, etc.

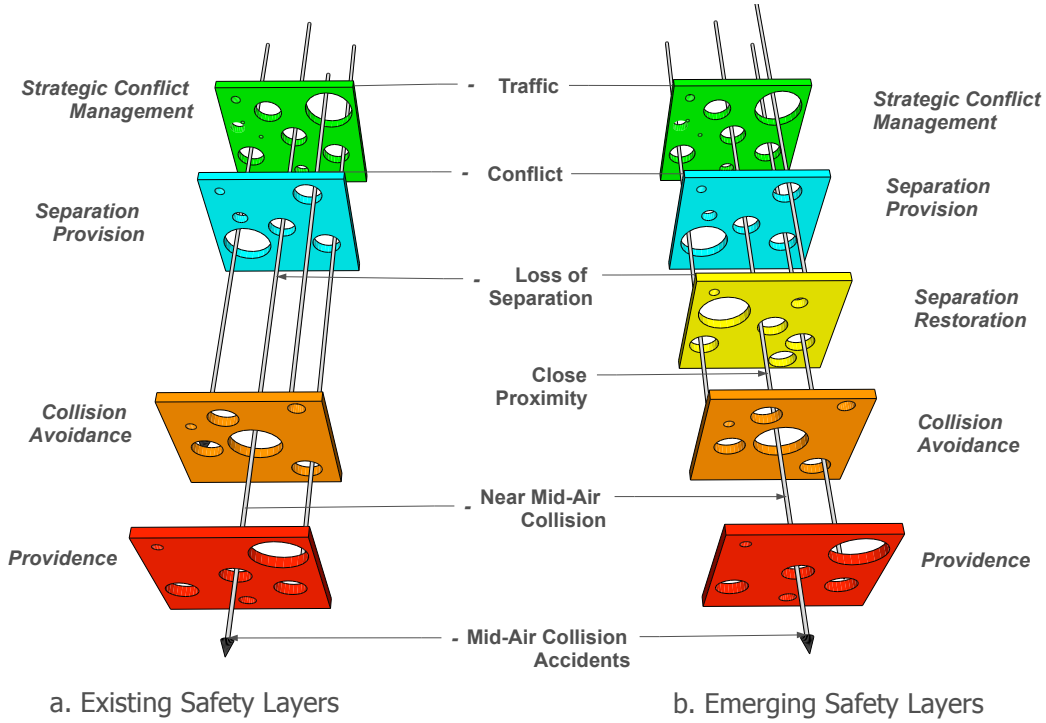



Figure 2.2: The safety layer models illustrating the layered approach used for collision avoidance in the ATM system.

of a pilot or a flight crew member. Note that, no specific definition of this safety event can be found in the existing regulations.[3] This event is included here to facilitate the discussion in this thesis (i.e. when the aircraft is not 'well clear' of other traffic, they are in *close proximity*) and to highlight the need for a separation standard for the definition of 'well clear' found in the regulation as addressed by Weibel *et al.* [2011].

- **Near Mid-Air Collision (NMAC)** is a situation in which the separation between the involving aircraft is simultaneously less than 500 ft horizontally and 100 ft vertically.

- **Mid-Air Collision (MAC)** is an aviation accident where two aircraft come into contact with each other while both are in flight.

[3]This concept is similar to but not equal to an **Airprox** defined in UK and to a **collision hazard** as mentioned in [Weibel *et al.*, 2011].

### 2.2.3 Risk Mitigation Mechanisms

Figure 2.3 shows the risk mitigation mechanisms used in the middle three layers of the emerging model, as shown in Figure 2.2b:

- **Separation Provision** layer includes all mitigation measures that keep aircraft away from hazards by at least the appropriate separation minima, by means of tactical intervention. It is currently carried out by Air Traffic Controllers (ATCOs) and, possibly in the future, by flight-crew with the assistance of on-board Airborne Separation Assurance System (ASAS), see Barhydt *et al.* [2003]; Hoekstra [2002].

- **Separation Restoration** layer includes all mitigation measures that prevent aircraft from operating in such proximity to other aircraft as to create a collision threat. One example of a mitigation measure is the Short Term Conflict Alert (STCA) system [Bakker, 2009], which is an automated warning system for ATCO; while another, by the flight-crew, is the visual separation manoeuvres that uncontrolled aircraft could make in order to remain well clear of other traffic, known as *self-separation* according to Zeitlin [2012].

- **Collision Avoidance** layer includes all mitigation measures that take all possible measures to ensure that an aircraft does not collide with any other aircraft. Depending on whether there exist datalinks between the involving aircraft, there are two types of mechanisms:

    1. **Cooperative**[1] collision avoidance mechanism includes all risk mitigation measures that rely on the information obtained from any communication links, such as transponders and ADS-B. For instance, Airborne Collision Avoidance System (ACAS) is an airborne safety net based on Secondary Surveillance Radar transponder signals.

    2. **Non-cooperative** collision avoidance is the lowest-level mechanism to prevent an imminent collision. In manned aviation, this entirely relies on the

[1]Note that, by cooperative, it means there is datalink, such as Automatic Dependent Surveillance-Broadcast (ADS-B), among the involving aircraft, it is not necessary that they are coordinating with each other to resolve a conflict.

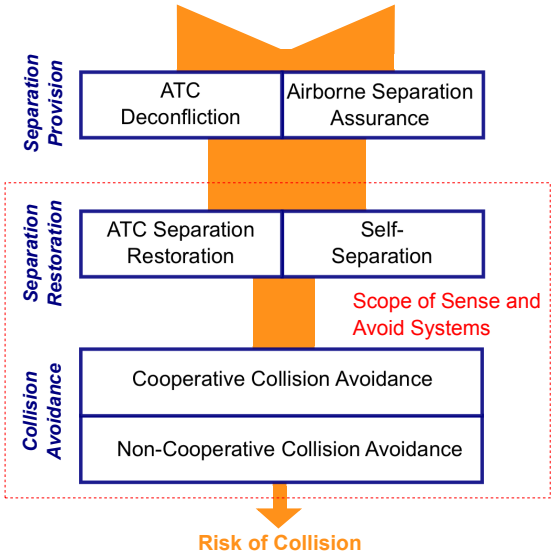



Figure 2.3: Risk mitigation mechanisms grouping into three safety layers. The scope of SAA is also highlighted.

> ability of the flight-crew to See and Avoid, in order to carry out the regulatory requirement specified in the Rules of the Air ICAO [2005a]: "Nothing in these rules shall relieve the pilot-in-command of an aircraft from the responsibility of taking such action, including collision avoidance manoeuvres based on resolution advisories provided by ACAS equipment, as will best avert collision."

## 2.3 Sense and Avoid Systems

Although the remote pilots of UAVs are not on-board the vehicles, none of the pilot's responsibilities in the existing collision risk mitigation measures should be relieved. For this purpose, UAS are required to be equipped with SAA system so as to enable the Unmanned Aerial Vehicles pilot (UAVp) to carry out these responsibilities. However, until the time of writing of this thesis, the standard for SAA systems is still under development, for instance, by EUROCAE (European Organisation for Civil Aviation Equipment) Working Group 73 and RTCA (Radio Technical Commission for Aeronautics) Special Committee 228. A preliminary Minimum Operational Performance

Standards (MOPS) establishing performance standards for UAS SAA systems in the operational environment specified in [RTCA, 2013] is expected to be available in July 2015.

Beside the standardization efforts, there exist technology demonstration programmes aiming at demonstrating the SAA capability with flight testing, for instance, the Mid Air Collision Avoidance System (MIDCAS) programme[1] across five European countries [Pellebergs, 2010], the ASTRAEA (Autonomous Systems Technology Related Airborne Evaluation & Assessment) programme[2] in UK [Hutchings *et al.*, 2007; Patchett & Ansell, 2010] and the Sense-and-Avoid Flight Tests (SAAFT) [Shakernia *et al.*, 2007] and ACAS $X_U$ [Petri & Spriesterbach, 2012] programmes in US.

### 2.3.1 A Safety-Analysis Development Process

Because of the potentially catastrophic consequences of error in the operation of SAA systems, the researchers from MITRE Corporation and Lincoln Laboratory, with their rich experience associated with the development and implementation of Traffic Alert and Collision Avoidance Systems II (TCAS II), have stressed the important role of a *safety-analysis methodology* in the development and standardization of such a safety-critical system, more than once in Kuchar [2005]; Zeitlin *et al.* [2006]; Zeitlin [2012]; Edwards [2012]; Cole *et al.* [2013].

Figure 2.4 shows a high level view of the safety-analysis methodology. This is an iterative development process, involving the following seven main steps:

1. Develop a **concept of operations** (CONOPS) to provide information, for example, on UAS flight characteristics, the environment in which the UAS will operate, responsibilities of the ground pilot, and communication protocols; and thus deriving the **initial requirements** for operations (where), functions (what), performance (how well) and safety (how risky), see, for example, the MIDCAS programme's CONOPS and requirement synthesis [Farjon & Sellem-Delmar, 2012; Sellem-Delmar, 2010];

2. Develop a **functional architecture** to accommodate the initial functional requirements, via identifying the necessary sub-functions and defining the data

[1] See the project website at http://www.midcas.org/.
[2] See the project website at http://astraea.aero/.

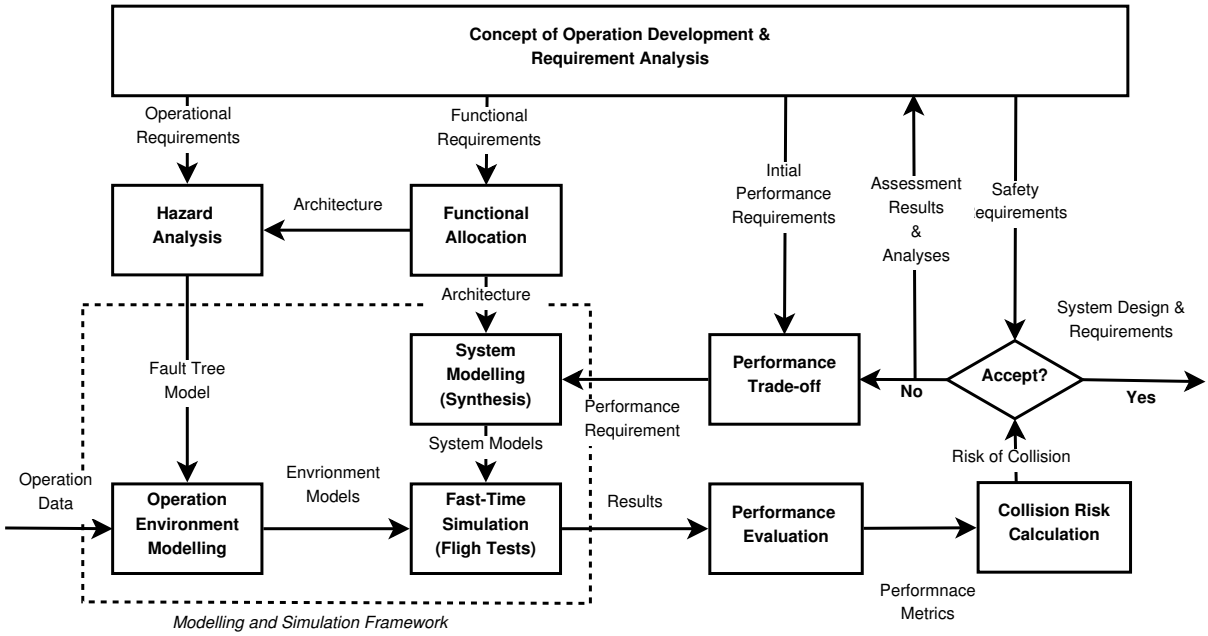



Figure 2.4: Safety-analysis methodology for the development of SAA systems.

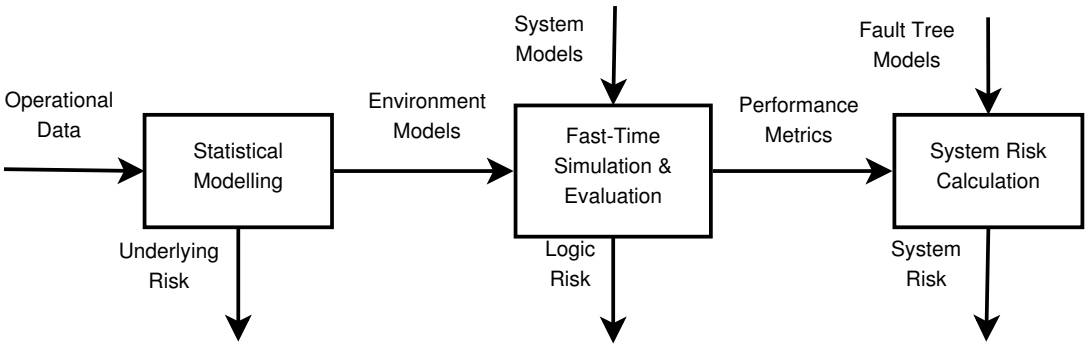



Figure 2.5: The process to calculate the system risk of collision, adapted from Raynaud & Arino [2006].

flow and interfaces among them, see §2.3.2 and 2.3.3 for more details;

3. Develop a **fault tree model**[1] to identify all events that could lead to a failure in the end-to-end SAA system, via analysing the potential failures of each function and data flow from the architecture step, see, for example, the MIDCAS's scenario assessment [Clarkson, 2012];

4. Develop a **modelling and simulation framework** to enable fast-time Monte Carlo simulations, so that sufficient data can be generated to statistically demonstrate the system effectiveness. Monte Carlo simulation is an established technique that repeatedly simulates the system models with independently selected values from their respective probability distributions. These simulations require:

   - a set of models to replicate the environmental and operational situations, e.g., encounter models [Kochenderfer *et al.*, 2008a, 2010b], sensor environment models [Griffith & Lee, 2011] and platform characteristics (vehicle dynamics, aircraft performance); and
   - the models for the SAA system candidates, including surveillance system, avoidance logics, and response characteristics (i.e. communications and pilot response latency).

5. Develop a recognized set of **performance metrics** to evaluate the system's safety performance, operational suitability and interoperability, see Edwards [2012]; Kochenderfer *et al.* [2010a]; ICAO [2007] for some useful metrics;

6. Calculate the **risk of collision**, using the tools developed in the previous steps and the process shown in Figure 2.5. A **risk assessment** can then be carried out to determine whether the resulting risk is at an acceptable level. If the risk is not acceptable, additional mitigation of risks will need to be considered in the CONOPS; or if the required performance is infeasible or at least undesirable for economic reasons, **performance trade-offs** will need to be made among the subfunctions so as to make certain requirements less onerous without compromising overall safety performance, see Zeitlin [2012]; and

[1] Also known as a *contingency tree* in some EUROCONTROL reports, such as Hutchinson & Drozdowski [2007] and Raynaud & Arino [2006, see pp. 10].

7. Validate the requirements with **flight testing** to reinforce the simulation results. The output of this overall process is the set of acceptable system design that satisfy the safety requirements.

### 2.3.2 Intended Functions and Sub-Functions

Although the standard functional requirements of SAA systems are still under development, it is expected that the SAA system would deliver at least two intended-functions[1] via eight sub-functions. According to the final report produced by FAA-Sponsored Workshop [2009, as cited in [George, 2012]], they are:

> *"Sense and Avoid (SAA) is the capability of a UAS to remain well clear from and avoid collisions with other airborne traffic. SAA provides the intended functions of self separation and collision avoidance as a means of compliance with the regulatory requirements to "see and avoid" compatible with expected behavior of aircraft operating in the airspace system. An SAA capability performs the following sub- functions (see Figure 2.6):*
>
> 1. *Detect - Determine presence of aircraft or other potential hazards;*
> 2. *Track - Estimated position and velocity (state) of a single intruder based on one or more surveillance reports;*
> 3. *Evaluate - Assess collision risk based on intruder and UA states;*
> 4. *Prioritize - Determine which intruder tracks have met a collision risk threshold;*
> 5. *Declare - Decide that action is needed;*
> 6. *Determine - Decide on what action is required;*
> 7. *Command - Communicate determined action; and*
> 8. *Execute - Respond to the commanded action."*

[1] Besides the two intended functions described here, MIDCAS proposes an additional function: provision of traffic information to allow the UAS pilot to build his situational awareness related to the surrounding traffics, [Farjon & Sellem-Delmar, 2012, see pp. 42].

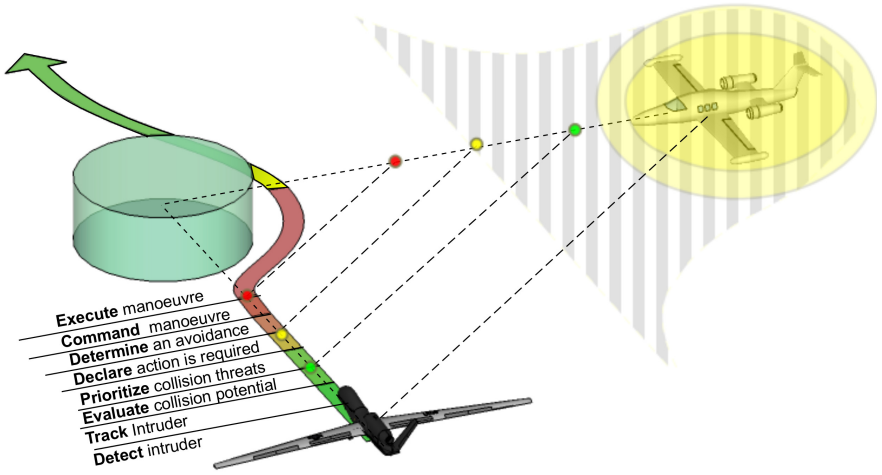



Figure 2.6: SAA Capability Sub-Functions, reproduced according to [FAA-Sponsored Workshop, 2009, as cited in [George, 2012]].

### 2.3.3 Functional Architecture

The design of each sub-system and the way they interact with each other constitutes the system architecture. Figure 2.7 shows a generic functional architecture used to accommodate the previous initial functional requirements, see Hutchings *et al.* [2007]; Patchett & Ansell [2010]; Dixon [2011].

The main functional elements are as follows:

- Surveillance Sensors: the environment must be first monitored using sensors and/or communications equipment. Surveillance sensors, depending on whether relying on datalinks, are divided into two groups: cooperative and non-cooperative. Table 2.1 summarizes the typical characteristics of four sensors technologies that are prevalent in the literatures [Griffith *et al.*, 2008; Chen *et al.*, 2011; Angelov, 2012; Boskovic *et al.*, 2013].

- Sensor Fusion and Tracking: the measurements from multiple sensor sources are fused together so as to improve the measurement accuracy and a tracker is required to associate successive measurements with specific targets. Once there are sufficiently consistent measurements for a specific target, a valid track is ini-

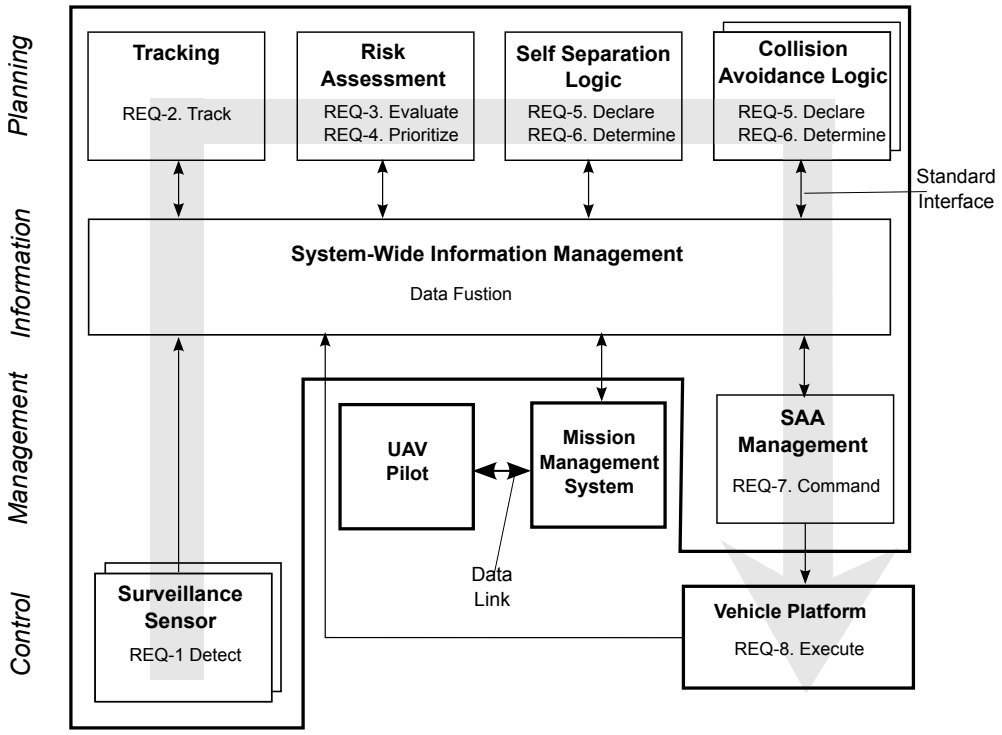



Figure 2.7: The functional architecture of a generic SAA system, for autonomous UASs, summarized from Hutchings *et al.* [2007]; Patchett & Ansell [2010]; Dixon [2011].

Table 2.1: Typical SAA sensor characteristics [Chen *et al.*, 2011].

| | | Cooperative Sensing | | Non-Cooperative Sensing | |
|---|---|---|---|---|---|
| | Unit | ACAS | ADS-B | Radar | EO |
| **Accuracy** in: | | | | | |
| Range | ft | 175 – 300 | - | 10 – 200 | - |
| Range rate | ft/s | - | - | 1–10 | - |
| Bearing | deg | 9 – 15 | - | 0.5 – 2 | 0.1 – 0.5 |
| Altitude | ft | 50 – 100 | 50 – 100 | - | - |
| Horizontal position | ft | - | 25 – 250 | - | - |
| **Update rate** | Hz | 1 | 1 | 0.2 – 5 | 20 |
| **Detection range** | nm | ≥ 14 | ≥ 20 | 5–10 | 2–5 |

tialized and, thereafter, will be updated over time. The tracking function should be capable of maintaining the track for a certain time even in the absence of a measurement update and capable of dropping the track after too many updates are missed, see, for example, [Fasano *et al.*, 2009; Chen *et al.*, 2011].

- Risk Assessment: each tracked object is assessed to decide if the track can be projected into the future with sufficient confidence and, if so, to evaluate the projected state against a set of criteria that would indicate the potentials for the track to become a close proximity or an NMAC. The tracked objects can then be prioritized according to the evaluated criteria. This allows the subsequent process to focus on the most critical objects with the limited on-board resource.

- Logics: given the prioritized list of tracked objects, both the Self Separation (SS) and Collision Avoidance (CA) logics are run in parallel, to detect, respectively, 1) any potential *close proximity* situations and 2) any potential NMAC situations. As discussed in the next section, the conflict detection and resolution (CDR) mechanisms, embedded in the logics, will then determine when to issue an alert and which advisory to issue. One of the main design challenges in SS is the lack of a recognized analytical definition of 'well clear', against which the SS logic should protect, [Weibel *et al.*, 2011]. Discussion on collision avoidance logics will be deferred to the next section.

- Management: given the SAA picture (consisting of the tracked objects, threat alerts, and resolution advisories), a decision-making mechanism among the UAVp, Mission Management System, and SAA Management should be in place to select the appropriate action for execution. For instance, as proposed by Hutchings *et al.* [2007], the UAVp can accept or reject the SS advisories, and can choose to either accept or ignore the CA manoeuvres until the SAA system is forced to take autonomous action, while the UAVp is always legally responsible, even during autonomous operation, for the safety of the UAV. However, there remains several open research questions here: the level of pilot involvement considering the possible communication delays and the design of a Human-Machine Interface (HMI) to provide the UAVp with sufficient situational awareness [Tadema, 2011].

- Vehicle Platform: the platform (including the Autopilot, Flight Control System and Navigation System) is responsible for the execution of the commanded manoeuvre and provision of the navigation states.

The focus of this work lies mainly in the Collision Avoidance Logic block and this will be discussed in detail in the subsequent sections.

## 2.4 Collision Avoidance Logics

### 2.4.1 Conflict Detection and Resolution

CDR in aviation is a methodology for maintaining separation between aircraft. This has been served as the underpinning principle for a wide range of systems to improve the safety level of the overall ATM. These systems include:

- the ground-based systems intended to assist ATCOs in:
  - strategically, maintaining separation among many aircraft [Jardin, 2003];
  - tactically, preventing close proximity between aircraft [Bakker, 2009].
- the airborne systems intended to assist fly crew in:
  - maintaining its separation with other aircraft [Barhydt *et al.*, 2003];
  - preventing Mid-Air Collision (MAC) against other aircraft in proximity [ICAO, 2007].

Depending on the operational context, these systems are generally referred to as CDR systems for separation provision in the Air Traffic Control (ATC) context, ASAS for separation provision and ACAS for collision avoidance in the avionics context.

Figure 2.8 shows all the necessary elements of the CDR process based on that described by Kuchar & Yang [2000]:

1. a response model to project the host aircraft's nominal/resolution trajectory;
2. a dynamic model to project the intruder aircraft's future state;

3. some criteria models used to evaluate the feasibility, and optionally the optimality, of the projected trajectories;
4. a mechanism to determine if an alert is required to be issued;
5. an action space to define all the available resolution actions; and
6. a mechanism to determine the resolution action to perform.

It is worth noting that, as mentioned in Kuchar & Yang [2000, pp. 181]:

- although the Criteria Models in Figure 2.8 is shown as a single block, different sets of decision criteria may be used for conflict detection and conflict resolution.
- as depicted by the dashed arrow in Figure 2.8, it is sometimes not clear how to separate conflict detection from conflict resolution, especially for the systems requiring a very low nuisance rate. For example, deciding when the action is required may depend on the type of action that will be performed; and similarly, the type of action that is required may depend on how early that actions begins. See the trigger mechanisms proposed by Patel & Goulart [2010] for an example of this interaction.

In his review paper, Kuchar & Yang [2000] presented a survey of 68 different methods to address CDR, most of which were driven by the desire to implement the CDR tools to assist the human operators in handling the expanding traffic loads and improve flow efficiency. In the last decade, along with the emerging need in developing the SAA systems, the diversity and quantity of CDR methods has been ever-increasing. Kopřiva *et al.* [2012] presented a survey of about 120 methods. While these methods cover a broad spectrum of operational context from the separation provision to collision avoidance safety layer, this work focuses only on the methods that are suitable for non-cooperative collision avoidance problems. Figure 2.9 relates the methods used in this thesis to the method categories proposed by the two survey papers.

### 2.4.2 Challenges and State-of-the-Arts

The recent development of collision avoidance logics are mainly driven by the following challenges: handling of non-cooperative intruders and robustness to uncertain intruders.

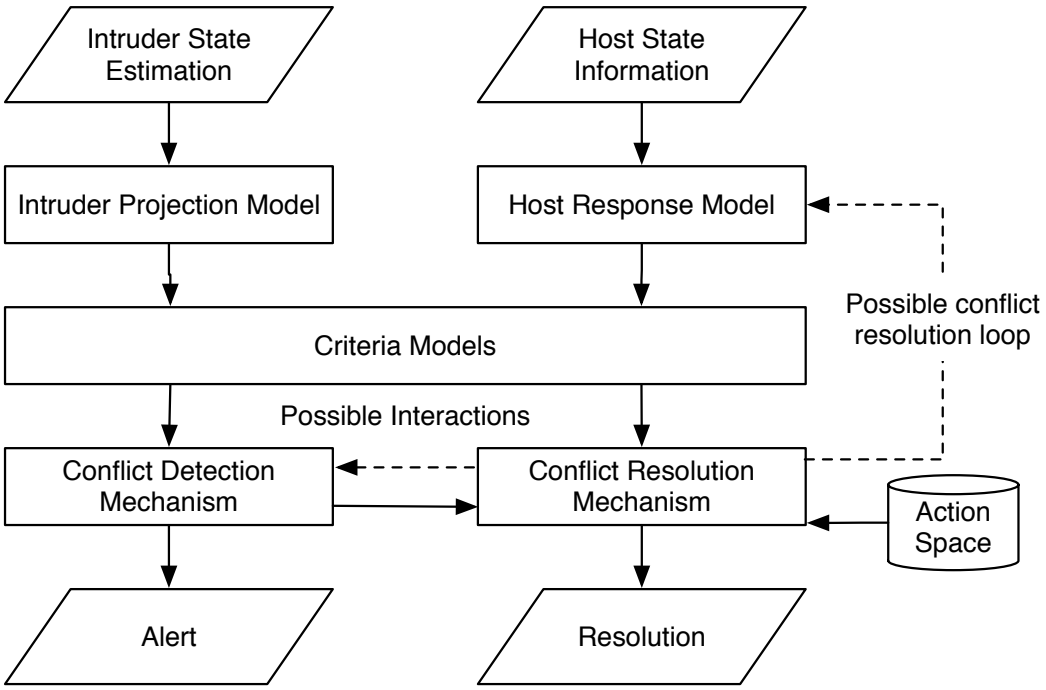
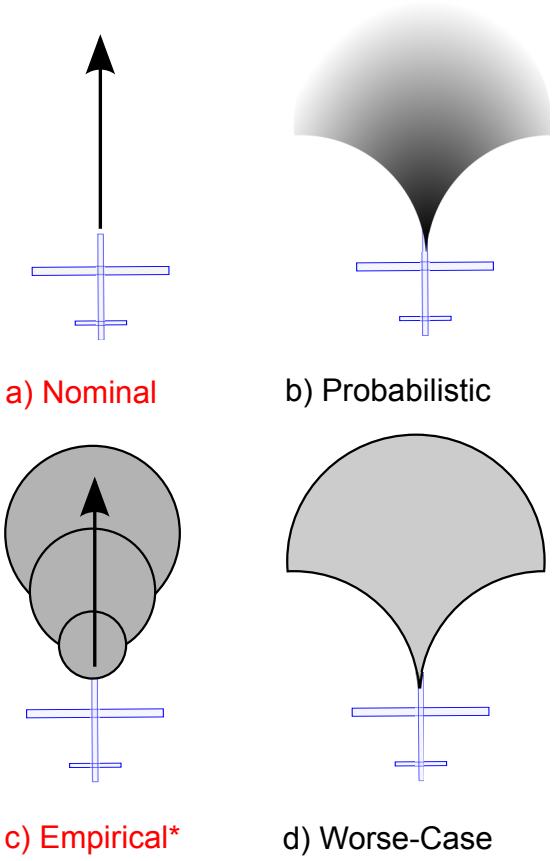
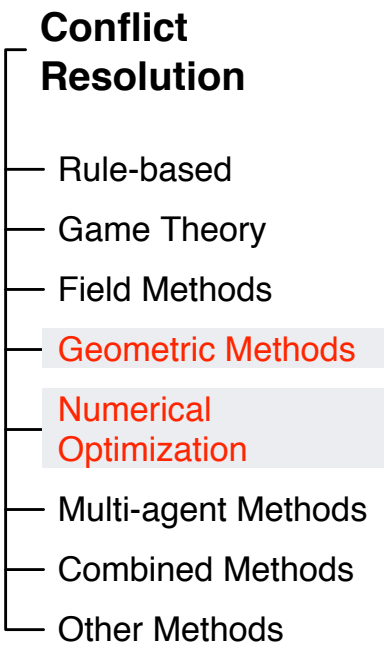



Figure 2.8: The necessary elements of a CDR process, which is the underpinning principle for all collision avoidance logics.




(a) Intruder state projection method category proposed by Kuchar & Yang [2000]. The empirical method (i.e. linear projection with growing uncertainty bounds) used in this thesis is inspired by Berry *et al.* [2010].




(b) Conflict resolution method category proposed by Kopřiva *et al.* [2012].

Figure 2.9: The relationship between the methods (for intruder state projections and conflcit resolution) used in this thesis to the existing methods in the literature.

### Collision Avoidance with Non-Cooperative Intruders

Not all intruder aircraft (or other airborne objects) are equipped to communicate their state information with other airspace users. When there is no communication link between the intruder and host aircraft, the host aircraft can only detect the airborne objects with on-board non-cooperative sensors, such as radar and Electro-Optical (EO)/Infra-Red (IR) sensors. Non-cooperative sensing normally results in a relatively less accurate state estimate, has shorter detection range and is subject to the limited FOR. All these limitations introduces additional technical challenges to collision avoidance with non-cooperative intruders.

In order to alleviate the effect of the limited FOR, additional measures have been studied. Fasano *et al.* [2008] introduced an additional "Blind Avoidance Manoeuvre" mode, in which the intruder's trajectory is estimated by the propagation of the last measured speed vector. Saunders & Beard [2008] proposed a nonlinear guidance law that attempts to manoeuvre the UAV in such a way that the intruder is moved to the edge of the Field of View (FoV), and thus guarantees that the UAV trajectory is not on the collision course with the obstacle.

Moreover, non-cooperative sensing approaches are divided into active and passive ones, depending on whether transmitting energy as part of the sensing. While the active sensors, including the component to generate the transmitted energy, are normally more expensive in terms of size, weight and power, not all UAVs are capable of carrying an active sensor system. Therefore, collision avoidance logics that are only based on passive sensors have been developed. For instance, Lai *et al.* [2012] and Cho *et al.* [2012] presented the collision avoidance logics based only on computer vision. Angelov *et al.* [2008] proposed a new passive approach for collision avoidance, which takes as input only the bearing between the host and intruder aircraft.

### Robustness to Uncertain Intruders

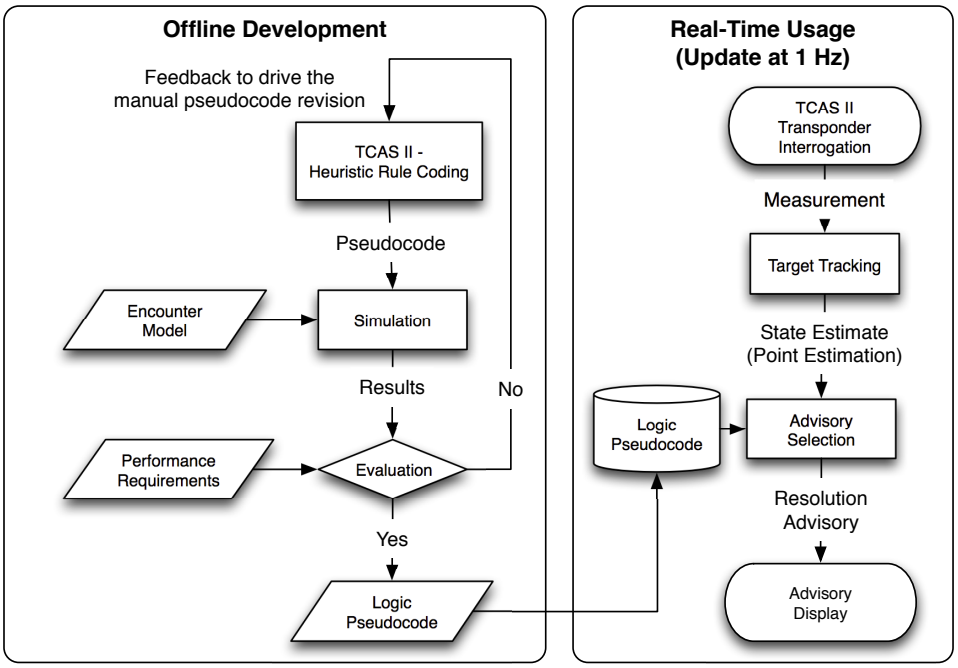
The estimate of the intruder's current state and future trajectory are inherently uncertain due to the limited accuracy of surveillance sensors and unknown intention of intruders. As the uncertainty propagates along the time, small uncertainties can significantly result in large estimate errors in the future. These estimation errors could trigger a nuisance alert in the detection phase or invalidate a selected resolution manoeuvre

in the resolution phase. Therefore, the treatment to the intruder uncertainty is a critical part of a collision avoidance logic. According to the method used to project the intruder's state, different measures have been used to handle the intruder uncertainty.

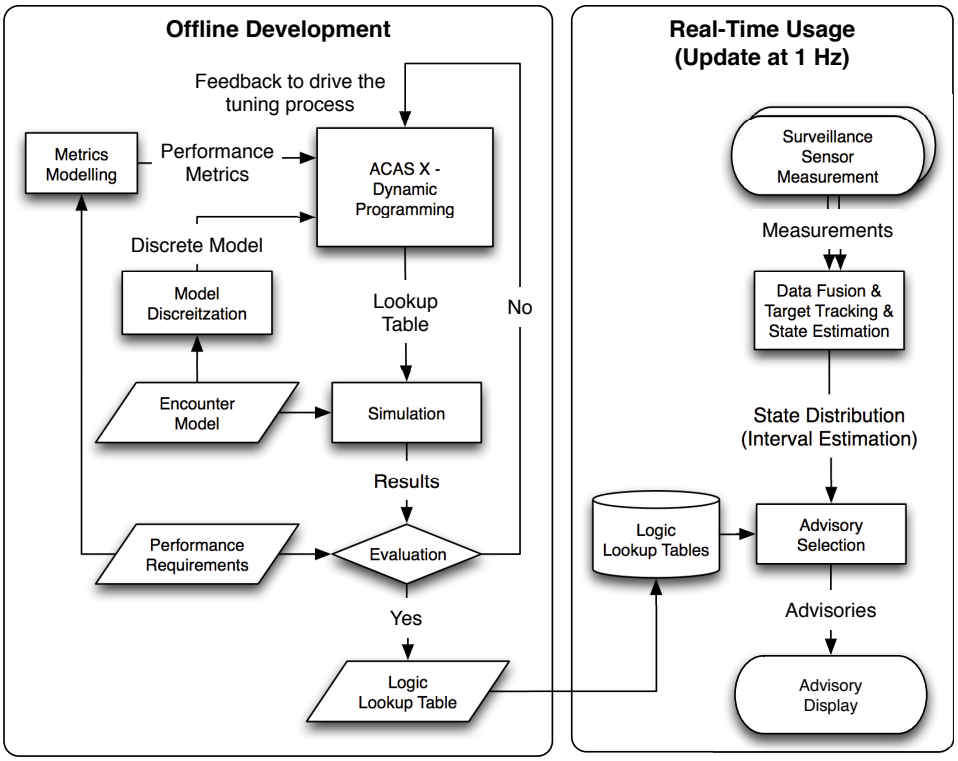
In the nominal methods, the current state is projected into the future along a single trajectory of maximum likelihood. The uncertainties are then handled by measures, such as introducing some safety buffer to accommodate the possible position error, introducing some time threshold to limit the extent the uncertainty could propagate, and introducing high-rate feedback to correct for the error. These measures are relatively simple and straightforward and have been applied to the existing Traffic Alert and Collision Avoidance Systems (TCAS) logic [ICAO, 2007], and other geometric collision avoidance logics, such as those of Carbone *et al.* [2006] and Shin *et al.* [2012].

In the probabilistic methods, the uncertainties are modelled as probability distributions to describe potential variations in the estimate, and the detection and resolution decisions are then made based on these probability distributions. For instance, Kochenderfer *et al.* [2012] models the intruder uncertainty as a Markov process and formulates a discrete-time stochastic optimal control problem. The problem is solved using *dynamic programming* to produce a numeric lookup table. The collision avoidance logic is encoded in the lookup table and will be used in real time via table lookups. The logic, used in the ACAS X as shown in Figure 2.10b, have two major features: the new model-based optimization development approach and the usage of interval-type estimation.

In the worse-case methods, it is assumed that the intruders' controls are uncertain but bounded within a certain range. The uncertainties are modelled by all possible manoeuvres within the bounded range. For instance, Bayen *et al.* [2003] applies a differential game formulation to a two-vehicle collision avoidance problem, where both vehicles are modelled as 2D kinematic models with constant speeds and bounded turn rates. A computational method, based on level set, is used to calculate the unsafe regions from which the intruder can cause a loss of separation with the host, regardless of the control action of host aircraft. These unsafe regions are used as a metric for conflict detection and resolution.




(a) Traditional development approach for TCAS II, the current generation of ACAS in operation.




(b) New development approach for the ACAS X, the next generation of ACAS.

Figure 2.10: Comparison of the development and usage processes of two generations of ACAS, based on Kochenderfer *et al.* [2011].

### 2.4.3 Problem Scope

To focus on the feasibility study of the application of high performance avoidance manoeuvres in collision avoidance logic, this thesis is confined to the collision avoidance problem with a single non-cooperative intruder and assumes that a surveillance and tracking module is available to provide a point estimation of intruder states.

However, it is worth noting that, the design of a collision avoidance logic should also take the following aspects, among others, into account:

- Multiple intruders: as the density of the airspace increases, encounters with multiple intruder aircraft will become increasingly likely. Especially, when an aircraft is performing an unplanned collision avoidance manoeuvre, it may encroach on an adjacent altitude level or cause secondary conflict with otherwise safely separated aircraft. Therefore, a collision avoidance logic should be able to handle multiple intruders. The proposed logic can be extended to accommodate multiple intruders by increasing the collision avoidance constraints in the problem formulation in §5.2.2, however, the effect of additional constraints on the logic performance will require further investigations.

- Interoperability: a new collision avoidance logic should be able to, whenever possible, interoperate with other existing collision avoidance systems. For instance, when encountering with a TCAS-equipped aircraft, the logic should be able to take the existing coordination mechanism into account, so as to prevent, for example, both aircraft from climbing into each other. The interoperability behaviour can be achieved by introducing an additional cost term to the problem formulation but this is left for future research.

## 2.5 Trajectory Planning

### 2.5.1 Trajectory Planning rather than Path Planning

As can be found in the survey paper on motion/trajectory planning algorithms by Goerzen *et al.* [2010] and the *Planning Algorithm* by LaValle [2006], the major difference between trajectory planning and path planning is the consideration of the differential constraints. Although *decoupled approaches*, or some other ad hoc measures, exist to

extend the path planning methods to handle the differential constraints, the high performance manoeuvres considered in this thesis (e.g. to constraint the aircraft's bank angle rather than its turn rate) would make implementing those extensions too challenging. Therefore, this thesis only considers the methods that can take the differential constraints into account.

### 2.5.2 Optimal Control Approach

According to the survey paper by Betts [1998], numerical methods for solving optimal control problems are divided into two major classes: indirect methods and direct methods. An indirect method attempts to solve the necessary conditions for optimal control and thus requires to explicitly derive the necessary conditions using *calculus of variation*, see Betts [2010, §4.1]. In contrast, a direct method does not require explicit derivation and construction of the necessary conditions. It directly transcribes, via parametrization and discretization, an Optimal Control Problem (OCP) to a finite-dimensional parameter optimization problem, which is then solved with an NLP algorithm. Among other reasons given by [Betts, 2010, §4.3], direct methods are more suitable for real-time implementation because they are less sensitive to the initial guess and thus have a larger radius of convergence.

With their great promise in real-time trajectory generation [Milam, 2003; Ross & Fahroo, 2006; Basset *et al.*, 2010; Drury *et al.*, 2010], direct methods have been used to generate obstacle-free trajectories for aerial vehicles. For example, Bollino & Lewis [2008] used a numerical solver called DIDO that is based on a pseudospectral direct method and sequential quadratic programming, to generate collision-free optimal trajectories for multiple UAVs. Flores [2007] developed a direct method based on the differential flatness property and the non-uniform rational B-spline basic functions. By utilizing the property of these basic functions, a dynamically feasible trajectory can be guaranteed to be generated within an obstacle-free corridor. Singla & Singh [2008] converted an obstacle avoidance problem to a convex one with a coordinate transformation and then solved it via a sequential linear programming approach. Patel *et al.* [2009] applied a direct multiple shooting method for an aircraft avoidance manoeuvre in the context of an anti-hijack system.

Among other direct methods reviewed by Hull [1997]; Betts [1998]; Rao [2009]

and Drury [2010, §2.4.3], the inverse-dynamic direct method described by Yakimenko [2000] is selected as the candidate method for further development because of its following features:

- a priori satisfaction of the boundary conditions;
- an absence of 'wild' trajectories, which do not satisfy the differential constraints, during optimization; and
- a small number of optimization variables.

In the case of computing short time aircraft manoeuvres, simulation results presented by Basset *et al.* [2010] have shown that these features would enable a good convergence robustness to the initial guess and a relatively fast computation time.

## 2.6 Summary

In order to confine the scope of this work, this chapter has presented the background information about the role of collision avoidance in the current ATM system and in SAA systems. Moreover, it has reviewed the latest work published in the field of collision avoidance logic development and found that the three state-of-the-art collision avoidance logics also lend themselves to the trajectory planning framework. Finally, it identified a potential algorithm for further development.

# Chapter 3

# Modelling and Simulation Framework

## 3.1 Introduction

In order to statistically demonstrate the effectiveness of collision avoidance systems of a safety-critical nature, large-scale Monte Carlo simulations are required to generate sufficient data for statistic analyses. For this capability, this chapter presents a modelling and simulation framework, which is constructed based on the Collision Avoidance System Safety Assessment Tool (CASSATT) developed by MIT Lincoln Laboratory, see Kochenderfer *et al.* [2010a]; Temizer [2011].

The framework is designed to be *modular* and *parametric*. It is modular in the sense that the whole framework is divided into different models according to the functional architecture given by Figure 2.7; and standard interfaces are used to connect these models. This enables rapid and simple adaptation as the design and/or fidelity of the models are evolving with the development. Moreover, the models are parametrized with a large set of configurable parameters (see Tables 3.7 and 3.8) to facilitate the parametric analyses for different purposes.

The framework was built in MATLAB and Simulink and the interfaces among the components are mainly according to the Interface Control Document (ICD) of the SAA system developed by BAE Systems [Dixon, 2011]. Figure 3.1 shows an overview of the framework. Given the test conditions:

1. the encounter model is used to generate initial conditions and nominal commands for both aircraft involved in a close encounter;

2. the nominal commands are then used to drive the host aircraft model and the intruder kinematic model;

3. the surveillance system model takes as input the current state of the intruder model and produces an estimate of the intruder state, termed target track;

4. based on the target track,[1] the collision avoidance logic determines whether a collision alert is required, and, if so, selects the resolution manoeuvre command;

5. finally, the host aircraft model executes the resolution/nominal commands and updates the navigation state. The process continues until the end of an encounter.

In the following, the detailed implementation of each component will be described and the interfaces and set of configurable parameters will be summarized in §3.7.

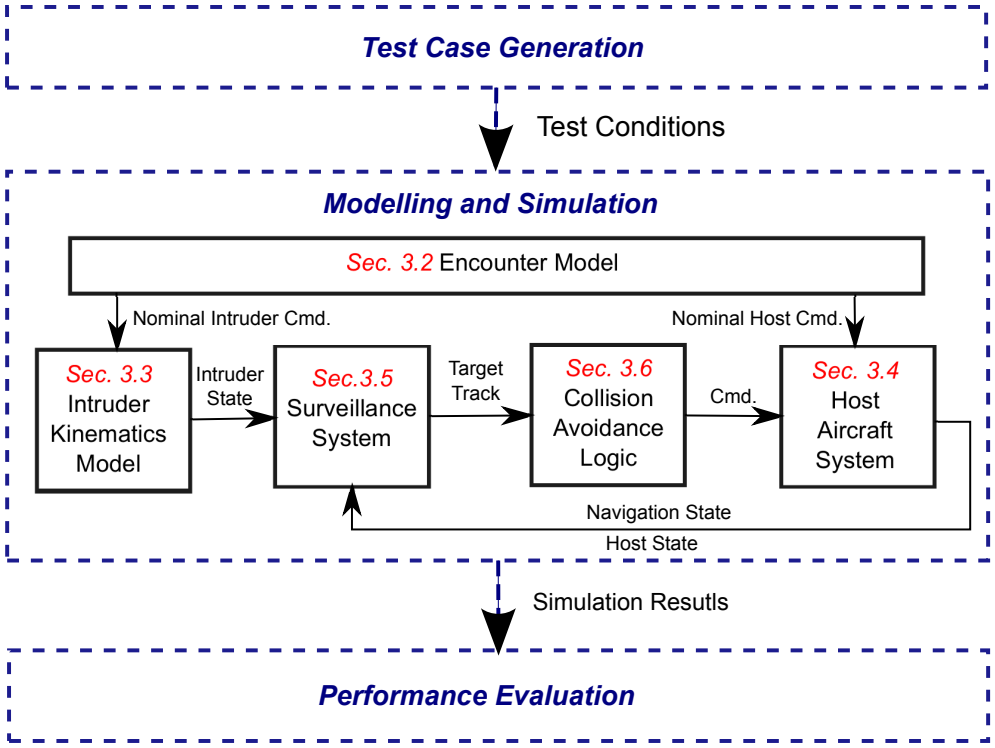



Figure 3.1: Modelling and Simulation Framework and Overview of Chapter 3.

[1]The actual input interface is a prioritized list of target track from a Risk Assessment process, shown in 2.7. This is required for multi-threat avoidance, but this work only focuses on single-threat cases.

## 3.2 Encounter Model

Briefly, an encounter model is a statistical model used to generate trajectories of the aircraft involved in a close-encounter traffic situation. This can be regarded as a replication of the airspace environment in which the collision avoidance system is being operated. Figure 3.2 shows the evolution of the existing encounter models, which are, respectively, developed by the International Civil Aviation Organization (ICAO) [see ICAO, 2007], European Organisation for the Safety of Air Navigation (EUROCONTROL) [see Arino *et al.*, 2002] and MIT Lincoln Laboratory [see Kochenderfer *et al.*, 2008a].

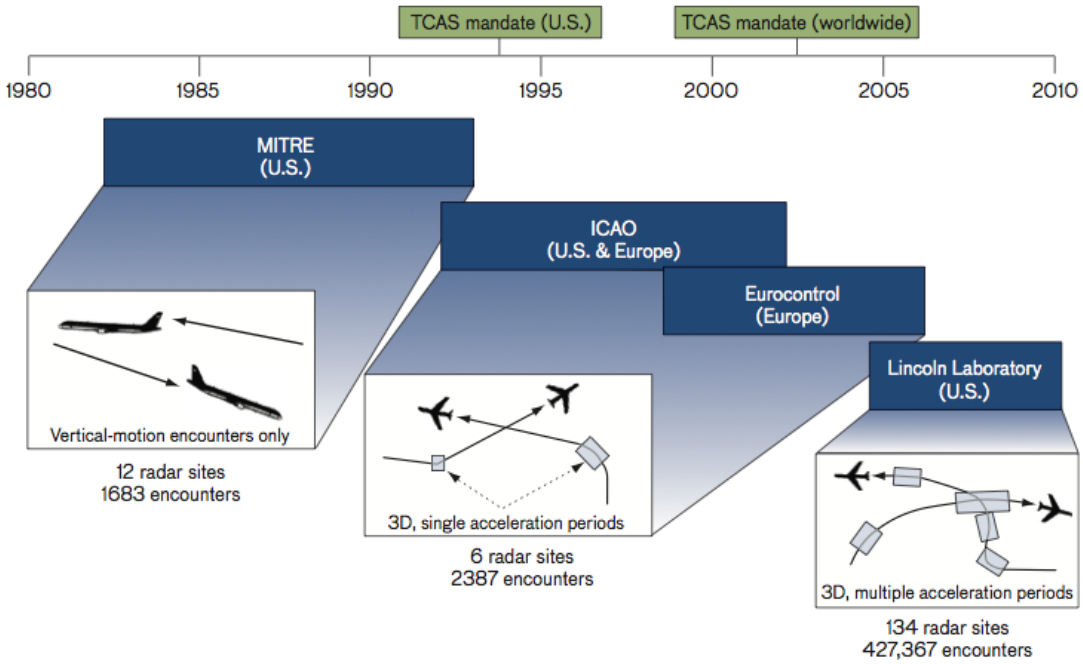



Figure 3.2: The evolution of the existing encounter models, adapted from Kochenderfer *et al.* [2008a].

Figure 3.3 shows the overview of an process to model and simulate an encounter model. The general process to construct an encounter model (§3.2.1) and the description of a particular Standard Encounter Model specified by ICAO [2007] (§3.2.2) will be presented. Using the Standard Encounter Model as an example, the process to generate and simulate an close encounter will be elaborated in §3.2.3 and §3.2.4.

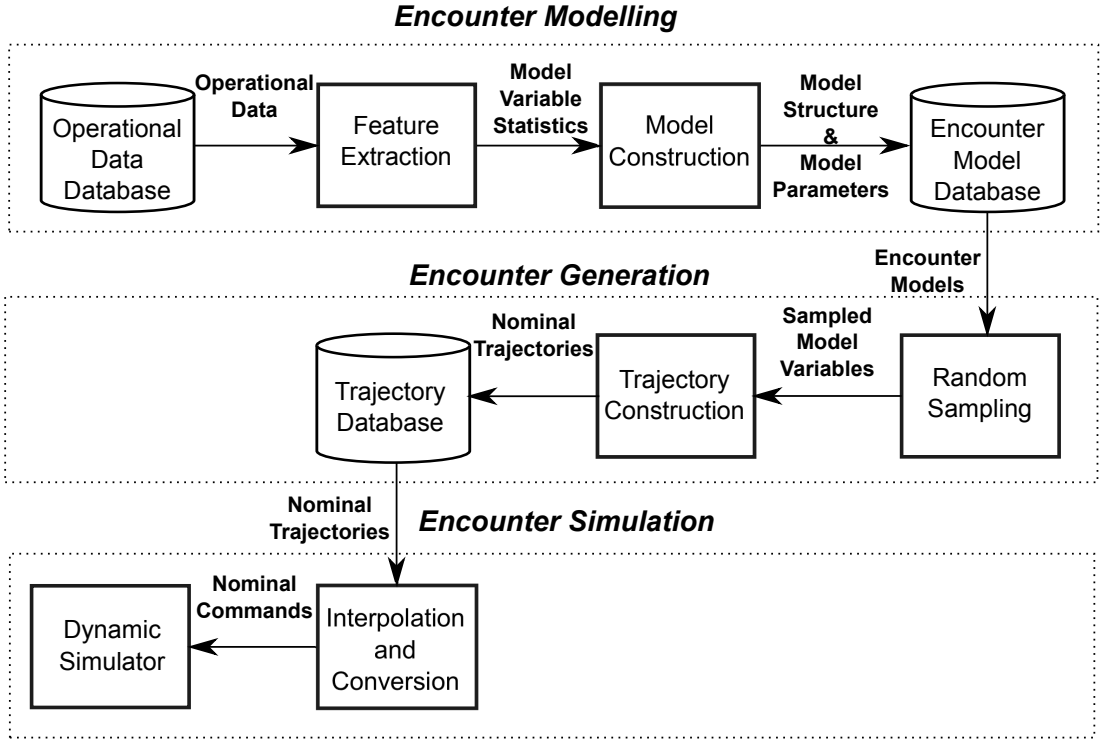



Figure 3.3: The outline of §3.2, showing the general process to construct and use an encounter model.

### 3.2.1 Encounter Modelling

**Encounter Types** Prior to extracting encounter features from the operational data, the *encounter type* of interest needs to be specified. According to Kochenderfer *et al.* [2008b,d] and Edwards *et al.* [2009], an encounter model can be one of three types:

1. A **correlated encounter model** is used to represent situations in which it is likely that there would be ATC intervention prior to a close encounter.

2. An **uncorrelated encounter model** is used to represent situations in which it is unlikely that there would be prior intervention by ATCO.

3. An **uncorrelated, unconventional encounter model** is used to represent the uncorrelated situations with an unconventional aircraft, i.e. an aircraft other than fixed-wing powered aircraft, such as a balloon or glider.

**Feature Extraction** Depending on the encounter type, the operational data needs to be filtered and processed in order to extract the features of the observed encounters.

Features may include static variables that specify an encounter (such as vertical or horizontal miss distance, approach angle, and altitude layer) and dynamic variables that describe the aircraft trajectories (such as turn rate and vertical rate at every second). These features will be represented by *model variables*, which will be elaborated along with the encounter model in § 3.2.2.

To aid in data processing, each feature was quantized into several bins and counts were taken of the frequency with which each bin was occupied by observed data. Based on these counts, probability tables were then constructed so that each feature can be randomly generated such that the overall geometries and dynamics are representative of the actual events observed in the data.

**Model Construction** With the model variables statistics, the model structure representing the interrelationship between the model variables needs to be identified. Figure 3.4 shows an example of such a model structure.

The example uses a *Bayesian network*[1] to represent four model variables and their dependency. The arrows show dependencies between variables, represented by conditional probability tables. For example, the probability of a given airspeed rate depends on airspace class, altitude, and airspeed. All these probability values in the conditional probability tables are the parameters of encounter models.

### 3.2.2 ICAO Standard Encounter Model

This subsection described the main elements of the Standard Encounter Model given by ICAO [2007], the full details of which can be found in ICAO [2007, §4.4.2.6]. This model was used to generate the nominal trajectories of close encounters in this work.

**Terminology** Figure 3.5 depicts an encounter window and some model variables specifying the encounter characteristic. The encounter window is a time interval only within which the nominal trajectories of the two aircraft in an encounter are defined.

[1] A Bayesian network model is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG).

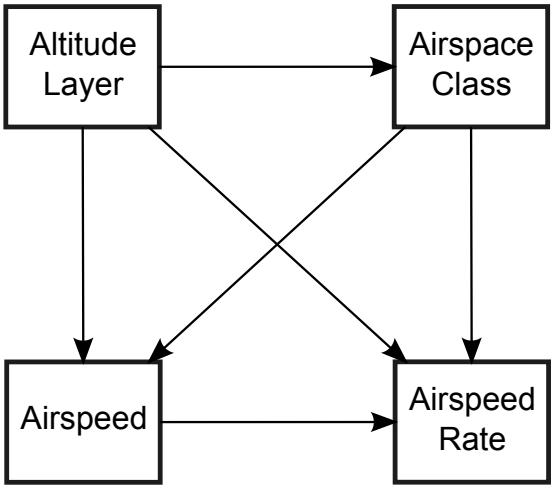



Figure 3.4: An example of the model structure of encounter models, using a Bayesian network.

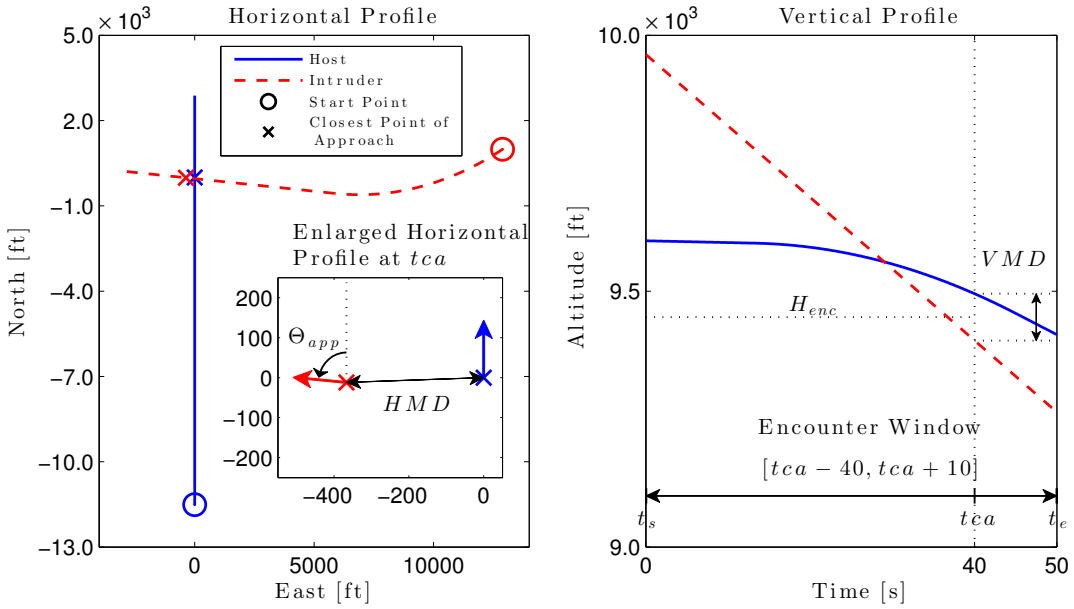



Figure 3.5: An example encounter illustrating the encounter window; and the encounter characteristics of time of closest approach ($tca$), approach angle ($\Theta_{app}$), horizontal miss distance ($HMD$), vertical miss distance ($VMD$), and encounter altitude ($H_{enc}$).

The time of closest approach, denoted by $tca$, is a *reference time*[2] at which various characteristics, including the vertical and horizontal miss distance ($VMD$ and $HMD$), are specified. Without loss of generality, the following reference times are used:

$$t_s = 0\text{s}; \quad tca = 40\text{s}; \quad t_e = 50\text{s}; \quad [t_s, t_e] = [0, 50]\text{s} \tag{3.2.1}$$

where $t_s$, $t_e$ are the start and end time of the encounter window $[t_s, t_e]$.

The encounter altitude, denoted by $H_{enc}$, is the average altitude of the two aircraft at the closest approach.

The approach angle, denoted by $\Theta_{app}$, is the the difference in the ground track angle of the two aircraft at closest approach, with 180 degrees defined as head on and 0 degrees defined as parallel.

**Encounter Sets** According to ICAO [2007, 4.4.2.6.1.1]:

> In order to calculate the effect of ACAS on the risk of collision and the compatibility of ACAS with ATM, sets of encounters shall be created for each of:
>
> 1. two aircraft address orderings;
> 2. six altitude layers (given in ICAO [2007, 4.4.1]);
> 3. nineteen encounter classes (given in ICAO [2007, 4.4.2.6.2.3.1]); and
> 4. nine or ten *vmd* bins (given in ICAO [2007, 4.4.2.6.2.4]).

However, for the preliminary evaluation in this work, only the encounter sets with the following properties are considered:

1. one altitude layer, $H_{enc} \in [5000, 10000]$ft;
2. nineteen encounter classes;
3. two *vmd* bins, $VMD \in [0, 100]$ ft for risk ratio and $VMD \in [0, 200]$ ft for nuisance alert rate.

This yields $38 = 1 \times 19 \times 2$ encounter sets in total.

[2]Note that, encounters in this model are constructed by building the trajectories of the two aircraft outwards starting at $tca$. When the process is complete, $tca$ may not be the precise time of closest approach and differences of a few seconds are acceptable.

**Model Variables** The geometric and kinematics properties of an encounter are determined by a set of random variables[1] in nature. These model variables, as depicted in Figure 3.5 and 3.6, include:

1. in the vertical plane, for all $i = \{1, 2\}$:
   (a) $VMD$: a vertical miss distance from the appropriate *vmd* bin;
   (b) $\dot{Z}_{0,i}$ and $\dot{Z}_{f,i}$: a vertical rate for each aircraft at the beginning of the encounter window and at the end of the encounter window;
   (c) $\ddot{Z}_i$: a vertical acceleration for each aircraft;
   (d) $T_{\ddot{Z},i}$: a start time for the vertical acceleration for each aircraft;
2. and in the horizontal plane, for all $i = \{1, 2\}$:
   (a) $HMD$: a horizontal miss distance;
   (b) $\Theta_{app}$: an approach angle;
   (c) $V_{CPA,i}$: a speed for each aircraft at closest approach;
   (d) $\delta_{turn,i}$: a decision for each aircraft whether or not it turns;
   (e) $\Theta_{trun,i}$, $\Theta_{bank,i}$ and $T_{trunEnd,i}$: the turn extent, bank angle and turn end time for each aircraft;
   (f) $\delta_{acc,i}$: a decision for each aircraft whether or not its speed changes; and
   (g) $A_{H,i}$: the magnitude of the speed change.

### 3.2.3 Encounter Generation

For each of the 38 encounter sets, 100 encounters are independently and randomly generated. The resulting 3800 encounters were stored in an Encounter Database, and served as the basis set of encounters for performance evaluation. The generation process involves two steps: *Random Sampling* and *Aircraft Trajectory Construction*:

[1] A random variable is assigned to each of the encounter characteristics. These random variables are denoted by upper case letters; and their realizations (the samples of a random variable) are denoted by the corresponding lower case letters with the subscripted index indicating the sample order.

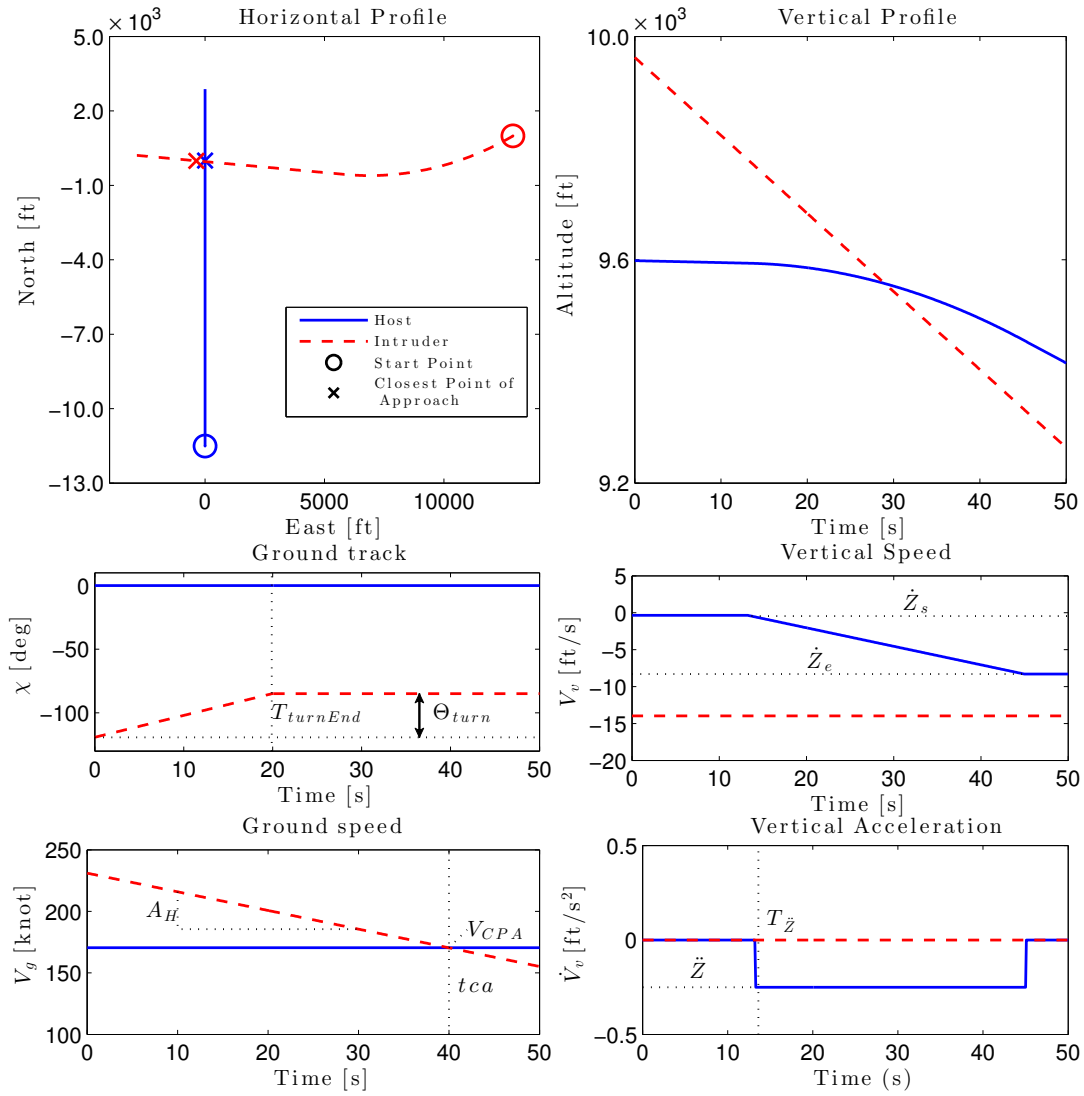



Figure 3.6: An example encounter to illustrate the model variables that specify the trajectory characteristics.

**Random Sampling** Along with the definitions of the model variables (specifying the encounter and trajectory characteristics), an encounter model would also give, as the model parameters, the probability distributions of the associated random variables. These probability distributions are normally given in the form of probability tables, either representing the random variable's probability density (mass) functions or cumulative distribution functions, see ICAO [2007, p. 4-37] and Kochenderfer *et al.*

Table 3.1: Part of the model variables specifying the encounter characteristics.

| Variable | Symbol | Distribution | | Unit |
|---|---|---|---|---|
| | | for risk ratio | for nuisance alert rate | |
| Encounter altitude | $H_{enc}$ | $\mathcal{U}(5000, 10000)$ | $\mathcal{U}(5000, 10000)$ | ft |
| Vertical miss distance | $VMD$ | $\mathcal{U}(0, 100)$ | $\mathcal{U}(0, 200)$ | ft |
| Horizontal miss distance | $HMD$ | $\mathcal{U}(0, 500)$ | $\mathcal{F}_{HMD}(x), x \in [0, 18228]$ | ft |
| Approach angle | $\Theta_{app}$ | $\mathcal{F}_{\Theta_{app}}(x), x \in [0, 180]$ | $\mathcal{F}_{\Theta_{app}}(x), x \in [0, 180]$ | deg |

[1] $\mathcal{U}$(a, b) is the uniform distribution with parameters $a$ and $b$.
[2] $\mathcal{F}_X(x)$ = Pr(X ≤ x) is the cumulative distribution function for a random variable $X$; the corresponding model parameters can be found in [ICAO, 2007, p. 4-40].

[2008c]. Table 3.1 shows, for example, the probability distribution of some model variables used in this work.

Furthermore, in order to model the dependencies between the model variables, their joint and conditional probability distribution are also given; for instance, Table 3.2 shows the conditional probability distribution table for the probability of a turn and a speed change.

Table 3.2: Conditional probability table of a turn and speed change.

| Layer | Pr(turn) | Pr(speed change\| a turn) | Pr(speed change\| no turn) |
|---|---|---|---|
| 1 | 0.31 | 0.20 | 0.5 |
| 2 | 0.29 | 0.20 | 0.25 |
| 3 | 0.22 | 0.10 | 0.15 |
| 4, 5, 6 | 0.16 | 0.05 | 0.10 |

With the probability distributions, any numbers of model variables can be obtained by *random sampling*. The sampled model variables can then be used to construct the trajectories for a nominal encounter. However, due to the limited aircraft performance of the host aircraft selected for this work, not all of these variables, specifying the trajectory characteristics, are achievable. For instance, according to ICAO [2007, p.4-37], the vertical rate $Z_{0,i}$ can take values between −6000 ft/min to 6000 ft/min, which is not achievable for the selected aircraft type. Therefore, a *rejection sampling* process is further required to ensure that the sampled model variables can be used to generate realistic encounters.

**Aircraft Trajectory Construction** Although the sampled model variables specify how the aircraft will manoeuvre during an encounter and their nominal relative positioning at $tca$, the nominal aircraft trajectories still remain unknown. A process is required to construct the trajectories that would match the given trajectory characteristics and result in the given encounter characteristics.

In order to satisfy the encounter characteristics at $tca$, the aircraft trajectories are building outwards starting at time of closest approach. This is mainly achieved by integrations of the kinematics variables while having the sampled model variables as the boundary values. For instance, as shown in Figure 3.6, the vertical acceleration $\dot{V}_v(t)$ is integrated to obtain the vertical speed $V_v(t)$ while having the sampled variables, $\dot{Z}_s$ and $\dot{Z}_e$, as the boundary values. In general, the process takes the following steps:

1. Initialization, see Figure 3.7:
   (a) establish a local Navigation coordinate system with its origin at an arbitrary position on the Earth's surface and its axes oriented in the north, east and down directions;
   (b) initialize the host such that it is located at the origin and flying due North;
   (c) initialize the intruder according to the sampled approach angle and Horizontal Miss Distance (HMD), [see Kochenderfer *et al.*, 2008b, p.41]
2. Construction of the aircraft trajectory in the vertical plane, see Figure 3.6:
   (a) construct the vertical acceleration trajectory;
   (b) integrate the vertical acceleration twice to get the altitude trajectory; and
   (c) shift both the altitude trajectories according to the encounter altitude.
3. Construction of the aircraft trajectory in the horizontal plane, Figure 3.6:
   (a) construct the trajectories of the ground speed rate and ground track angle rate, according to the sampled variables given in page 34;
   (b) integrate the above two trajectories to obtain the trajectories of the ground speed and ground track angle;
   (c) calculate the north and east components of their velocities; and

(d) integrate the north and east speeds to get the aircraft trajectory in the horizontal plane.

Note that, it is possible for the selections made for the various characteristics of an encounter to be infeasible. When this occurs, the selection for a particular characteristic is re-sampled until the resulting encounter is feasible.

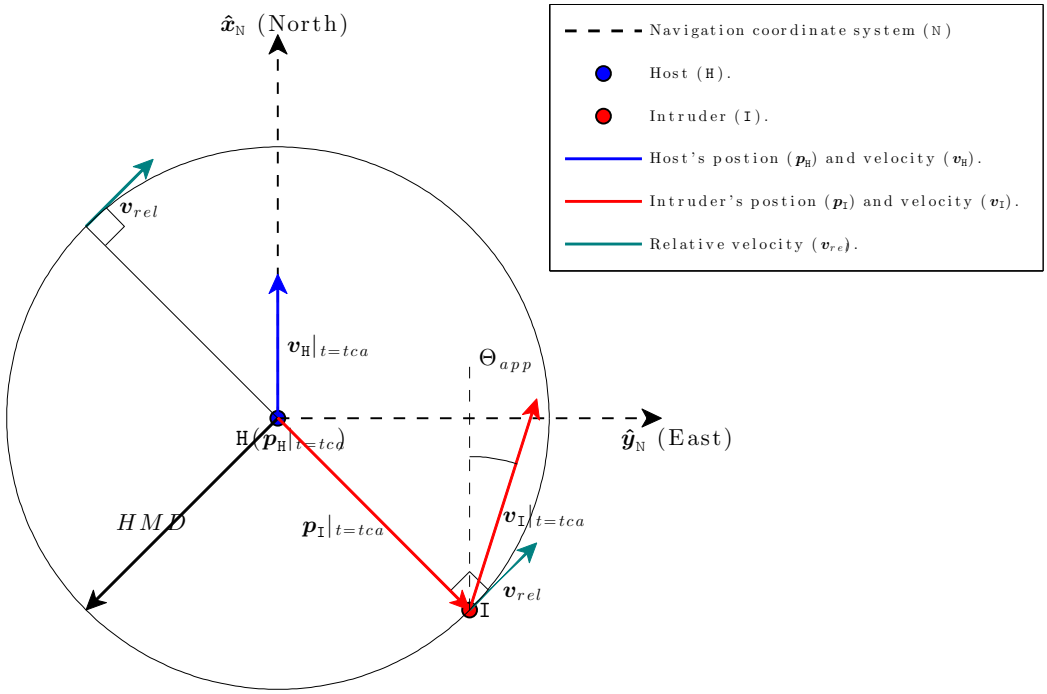



Figure 3.7: Establishment of the navigation coordinate system and initialization of the aircraft trajectory construction process.

The outputs of the trajectory construction process are the discrete time histories of both aircraft's positions and velocity expressed in the navigation coordinate system, with a time step $\Delta t = 0.1$ s:

$$\boldsymbol{p}_i(k\Delta t) = \begin{bmatrix} p_{i,N}(k\Delta t) \\ p_{i,E}(k\Delta t) \\ p_{i,D}(k\Delta t) \end{bmatrix}, \qquad \boldsymbol{v}_i(k\Delta t) = \begin{bmatrix} v_{i,N}(k\Delta t) \\ v_{i,E}(k\Delta t) \\ v_{i,D}(k\Delta t) \end{bmatrix},$$

$$\forall k = [0, 1, \ldots, 500] \quad \text{and} \quad \forall i \in \{\mathtt{I},\mathtt{H}\} \tag{3.2.2}$$

where point H and point I denote the host and intruder respectively.

The nominal trajectories for 3800 encounters are saved in an *encounter database* file that will be used in the simulation.

### 3.2.4 Encounter Simulation

During the simulation, the discrete nominal trajectories will be interpolated to give both aircraft's velocities $\boldsymbol{v}_i(t), \forall i \in \{\mathtt{I},\mathtt{H}\}$, which will then be transformed to the manoeuvre command vectors $\boldsymbol{c}$:

$$\boldsymbol{c}_i = [V_{g,i} \quad \chi_i \quad V_{v,i}]^T, \quad \forall i \in \{\mathtt{I},\mathtt{H}\} \tag{3.2.3}$$

with

$$\boldsymbol{c}_i = \begin{bmatrix} V_{g,i} \\ \chi_i \\ V_{v,i} \end{bmatrix} = \begin{bmatrix} \sqrt{(v_{i,N})^2 + (v_{i,E})^2} \\ \text{atan2}\,(v_{i,E}, v_{i,N}) \\ -v_{i,D} \end{bmatrix}, \tag{3.2.4}$$

where $V_g$, $\chi$ and $V_v$ are the ground speed, ground track angle and vertical speed, respectively.

## 3.3 Intruder Model

The intruder model takes as input the manoeuvre commands and produces the aircraft trajectory based on the basic kinematics. Figure 3.8 shows the architecture of the intruder model, in which the Command Disturbance block is used to introduce *manoeuvre uncertainty*; the Limiter block is to ensure the resulting intruder manoeuvre remains realistic; and the Coordinate Transformation and Integrator are used to propagate the intruder's position in the navigation coordinate system.

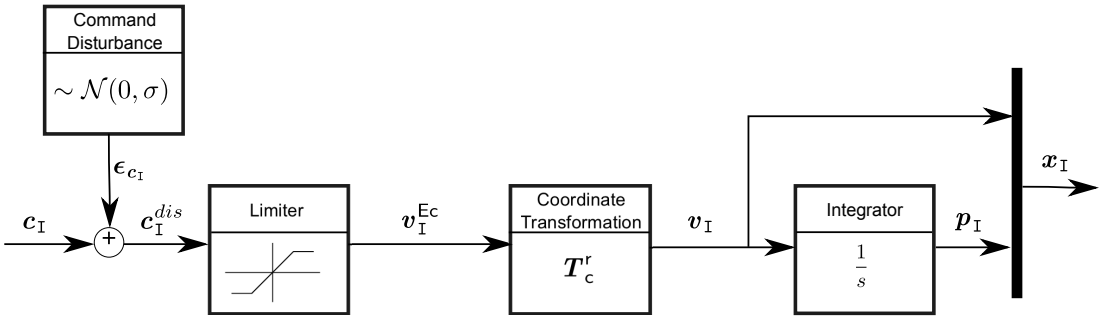



Figure 3.8: Intruder model architecture.

### 3.3.1 Manoeuvre Uncertainty

The intruder's manoeuvre uncertainty is modelled by an input disturbance. The levels of uncertainty can be controlled by a simulation switch, which will be set according to the testing conditions.

1. When the switch is set to zero, the command disturbance vector $\boldsymbol{\epsilon}_{\boldsymbol{c}_\mathrm{I}}$ is equal to zero:

$$^{0}\boldsymbol{\epsilon}_{\boldsymbol{c}_\mathrm{I}} = \mathbf{0}_{3\times 1} \tag{3.3.1}$$

2. When the switch is set to one, the components of the command disturbance vector $\boldsymbol{\epsilon}_{\boldsymbol{c}_\mathrm{I}}$ are normally distributed:

$$^{1}\boldsymbol{\epsilon}_{\boldsymbol{c}_\mathrm{I}} = [\epsilon_{V_g} \quad \epsilon_\chi \quad \epsilon_{V_v}]^T, \tag{3.3.2}$$

where

$$\epsilon_{V_g} \sim \mathcal{N}(0, \sigma_{V_g}), \quad \epsilon_\chi \sim \mathcal{N}(0, \sigma_\chi), \quad \epsilon_{V_v} \sim \mathcal{N}(0, \sigma_{V_v}),$$

with the variances $\sigma_{V_g}$, $\sigma_\chi$ and $\sigma_{V_v}$ are simulation parameters.

### 3.3.2 Intruder Limitations

In order to ensure the resulting intruder manoeuvre remains realistic, a limiter is used to impose the range and rate limits on the manoeuvre commands:

$$\boldsymbol{c}_\mathrm{I}^{dis} = \boldsymbol{c}_\mathrm{I} + \boldsymbol{\epsilon}_{\boldsymbol{c}_\mathrm{I}}. \tag{3.3.3}$$

Table 3.3: Intruder limitations parameters.

| Variable | Limitations | | Unit |
|---|---|---|---|
| | Maximum | Minimum | |
| $V_g$ | 400 | 0 | knots |
| $\dot{\chi}$ | 50 | -50 | deg/s |
| $V_v$ | 3000 | -3000 | ft/min |

The limit values for all the manoeuvre commands are summarized in Table 3.3.

### 3.3.3 State Equation

The state equation of the intruder model represents the vehicle kinematics:

$$\dot{p}_{\mathrm{I},N} = V_{g,\mathrm{I}} \cos \chi_{\mathrm{I}}, \tag{3.3.4a}$$

$$\dot{p}_{\mathrm{I},E} = V_{g,\mathrm{I}} \sin \chi_{\mathrm{I}}, \tag{3.3.4b}$$

$$\dot{p}_{\mathrm{I},D} = V_{v,\mathrm{I}}, \tag{3.3.4c}$$

and augmenting the intruder state vector with its velocity yields:

$$\boldsymbol{x}_{\mathrm{I}} = \begin{bmatrix} p_{\mathrm{I},N} & p_{\mathrm{I},E} & p_{\mathrm{I},D} & v_{\mathrm{I},N} & v_{\mathrm{I},E} & v_{\mathrm{I},D} \end{bmatrix}^T, \quad \text{or simply} \quad \boldsymbol{x}_{\mathrm{I}} = \begin{bmatrix} \boldsymbol{p}_{\mathrm{I}} \\ \boldsymbol{v}_{\mathrm{I}} \end{bmatrix}. \tag{3.3.5}$$

## 3.4 Aircraft System Model

Due to the fact that the sensor's FoV is a function of the aircraft's attitude, the aircraft model should be able to provide the attitude information according to the realistic aircraft dynamics; therefore, the following modelling objectives are required:

1. **Aircraft Performance Compliance:** The model should take the given aircraft performance into account.

2. **Attitude Estimate Provision:** The model should be able to provide the attitude estimates.

3. **Manoeuvre Command Tracking Performance Configurability:** The transient response characteristic of the manoeuvre command tracking should match that of the actual aircraft platform.

The model takes as inputs the manoeuvre commands and then generates the navigation state as output. Figure 3.9 shows an overview of the parametric aircraft system model. As can be seen from the dashed arrows in the figure, the model is parametrized with the aircraft's performance in terms of manoeuvre command tracking, flight dynamics, and navigation systems. The performance parameters can be configured to match those of the actual host aircraft platform being tested. The steady wind conditions can also be injected into the aircraft dynamics model. This flexibility is of particular importance due to the diverse performance of numerous types of UAVs.

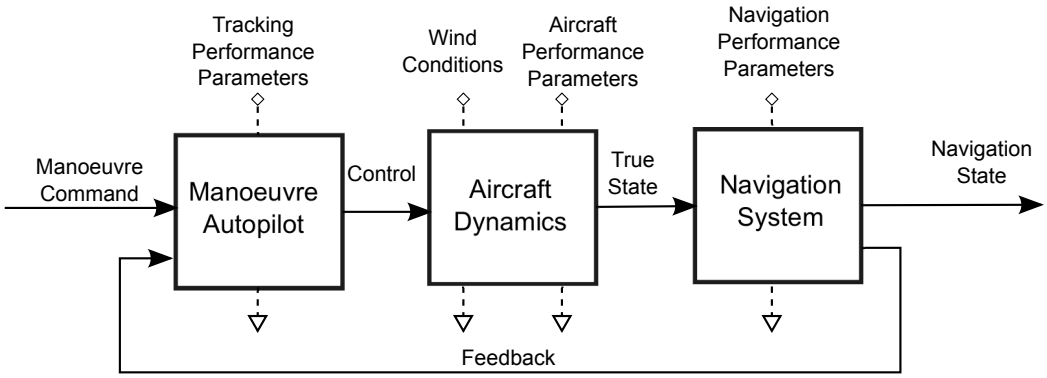



Figure 3.9: Overview of a configurable aircraft system model.

In response to the requirement of aircraft performance compliance, Section 3.4.1 describes an aircraft dynamics model, consisting of the equations of motion and aerodynamic force model described by Hull [2010], the propulsion force model given by Nuic [2012] and the generic performance limitations model.

In response to the requirement of attitude estimate provision, Section 3.4.2 describes the outputs of the navigation system model, which provides three groups of outputs for the purposes of output tracking, state feedback and navigation (including the attitude estimation), respectively.

In response to the requirement of manoeuvre command tracking performance configurability, Section 3.4.3 describes the implementation of a programmable manoeuvre autopilot, which is based on the Nonlinear Dynamic Inversion (NDI) concept, see

Ducard [2009, Ch. 6] and Stevens & Lewis [2003, Ch. 5.8] for more details. The idea behind the NDI scheme is to transform the nonlinear system into a linear one and then design a controller for the transformed linear system. The details about the transformation and the controller design are also presented.

Some example results are presented in Section B.2 to verify the modelling requirements.

### 3.4.1 Aircraft Dynamics

**Equation of Motion** A three Degree of Freedom (3-DoF) point-mass model is used to represent the flight dynamics and vehicle performance of the host aircraft. Figure 3.10 shows the coordinate system used for the derivation of the point-mass model.

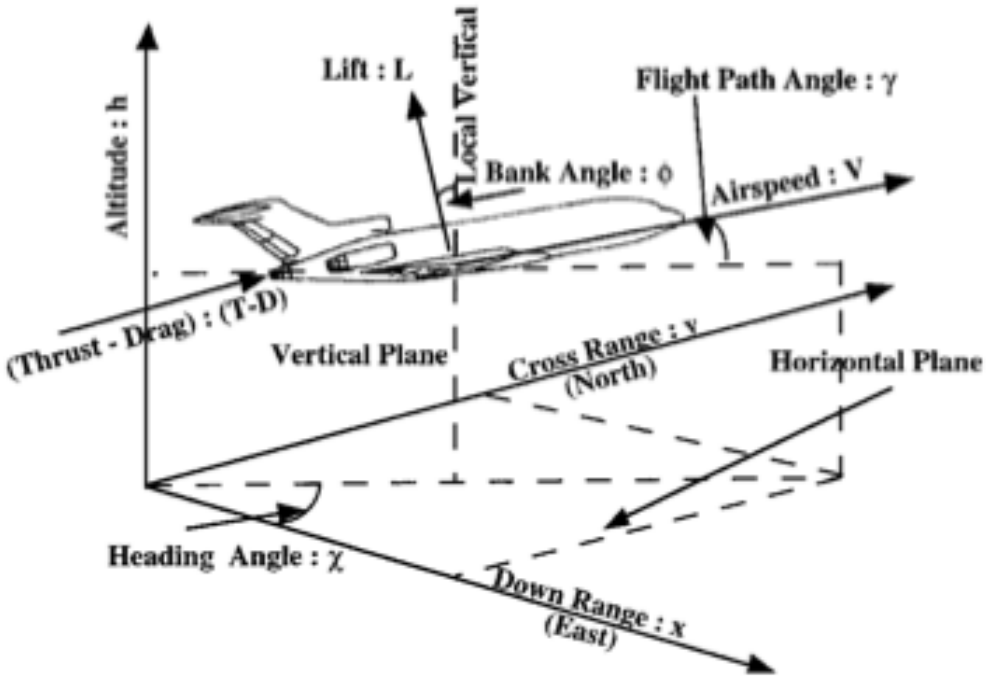



Figure 3.10: Parameter definitions of the aircraft point-mass model in the navigation coordinate system N, adapted from Menon *et al.* [1999]

For manoeuvres undertaken over the short period of collision avoidance, it can be assumed that:

1. the fuel expenditure is negligible, so the aircraft mass remains constant;
2. the wind is steady, i.e. time-invariant and uniform, so the wind condition remains constant;
3. the side-slip angle is zero, as only coordinated flight is considered;
4. the angle of attack is small, so the thrust is approximately aligned with the velocity.

With the above assumptions, the Equations of Motion (EoM) for a generic aircraft over the flat Earth can be written as:

$$\dot{p}_{\mathrm{H},N} = V_a \cos\gamma_a \cos\chi_a + v_{\mathrm{W},N} \tag{3.4.1a}$$

$$\dot{p}_{\mathrm{H},E} = V_a \cos\gamma_a \sin\chi_a + v_{\mathrm{W},E} \tag{3.4.1b}$$

$$\dot{p}_{\mathrm{H},D} = -V_a \sin\gamma_a + v_{\mathrm{W},D} \tag{3.4.1c}$$

$$\dot{V}_a = \frac{T - D}{m} - g\sin\gamma_a \tag{3.4.1d}$$

$$\dot{\chi}_a = \frac{g n_z \sin\mu}{V_a \cos\gamma_a} \tag{3.4.1e}$$

$$\dot{\gamma}_a = \frac{g(n_z \cos\mu - \cos\gamma_a)}{V_a} \tag{3.4.1f}$$

where:

$p_{\mathrm{H},N}$, $p_{\mathrm{H},E}$, $p_{\mathrm{H},D}$: are the North, East, Down components of $\boldsymbol{p}_{\mathrm{H}}$, the host's position in the navigation coordinate system.

$v_{\mathrm{W},N}$, $v_{\mathrm{W},E}$, $v_{\mathrm{W},D}$: are the North, East, Down components of $\boldsymbol{v}_{\mathrm{W}}$, the wind velocity in the navigation coordinate system.

$V_a$, $\chi_a$, $\gamma_a$: are the true airspeed, aerodynamic heading angle, and aerodynamic flight-path angle, respectively.

$\mu$: is the bank angle.

$n_z$: is the normal load factor, defined as $n_z = L/mg$.

$g$: is the acceleration due to gravity.

$m$: is the aircraft mass.

$T$, $L$, $D$: are the thrust, lift and drag respectively.

Note that, true airspeed $V_a$ is equal to the inertial speed $V$ when there is no wind; aerodynamic heading $\chi_a$ is equal to the aircraft heading $\psi$ when there is no side-slip $\beta$; and bank angle is equal to roll angle $\phi$ when angle of attack $\alpha$ and side-slip $\beta$ are both zero.

**Force Models** The aerodynamics and propulsion forces are aircraft-type-dependent; they are normally functions of aerodynamic angle, Mach number, altitude, control inputs and aircraft specific coefficients, see Stevens & Lewis [2003] and Hull [2010] for the detailed description of these functions; and see the Base of Aircraft Data (BADA), Nuic [2012], for the parameters of most currently operating aircraft.

For a particular type of aircraft, the thrust is modelled as a function of throttle setting $\bar{T}$, airspeed $V_a$, and altitude $h$:

$$T = f_T(\bar{T}, V_a, h) = \bar{T} T_{max} \tag{3.4.2}$$

with

$$T_{max} = C_{Tcr}\left(\frac{C_{Tc,1}}{V_a}\left(1 - \frac{h}{C_{Tc,2}}\right) + C_{Tc,3}\right) \tag{3.4.3}$$

where:

$h$: is the altitude and equals to the negative down position ($-p_{\mathrm{H},D}$).

$T_{max}$: is the maximum available thrust during the cruise phase.

$C_{Tc,i}$, $C_{Tcr}$: is the thrust coefficients, as defined in Nuic [2012, p.22].

The lift and drag are modelled as functions of angle of attack $\alpha$, airspeed $V_a$, and altitude $h$:

$$\begin{aligned} L &= f_L(\alpha, V_a, h) = \tfrac{1}{2}\rho V_a^2 S C_L \\ D &= f_D(\alpha, V_a, h) = \tfrac{1}{2}\rho V_a^2 S C_D \end{aligned} \tag{3.4.4}$$

with the lift and drag coefficient defined as:

$$C_L = C_{L0} + \alpha C_{L\alpha} \tag{3.4.5a}$$

$$C_D = C_{D0} + k C_L^2 \tag{3.4.5b}$$

where:

- $S$: is the aircraft wing area.
- $\rho$: is the air density, which is modelled as a function of altitude $\rho(h)$ according to the International Standard Atmosphere (ISA).
- $C_{L0}$, $C_{L\alpha}$, $C_{D0}$, $k$: are the zero-angle-of-attack lift coefficient, lift curve slope, zero-lift drag coefficient, and induced drag factor, respectively. All of them are functions of the Mach number $M$ and assumed to be available in the form of look-up tables.
- $M$: is the Mach number, a function of the true airspeed and altitude $M(V_a, h)$.

**Performance Limitations** In order to reflect the realistic aircraft behaviour, the aircraft performance is taken into account by restricting the corresponding variables to their upper and lower bounds. The *flight envelope* and the *V-n diagram* for the Jetstream 31 aircraft can be derived from the above force models, which are shown in Appendix B.1.

For the performance in propulsion systems and airframe aerodynamics:

$$0 \leq \bar{T} \leq 1, \quad \alpha_{min} \leq \alpha \leq \alpha_{max} \tag{3.4.6}$$

For the flight envelope:

$$0 \leq h \leq h_{max}, \quad V_{a,min} \leq V_a \leq V_{a,max} \tag{3.4.7}$$

For the rolling performance and the structural limitations on load factor:

$$\mu_{min} \leq \mu \leq \mu_{max}, \quad \dot{\mu}_{min} \leq \dot{\mu} \leq \dot{\mu}_{max}, \quad n_z \leq n_{z_{max}}^{str} \tag{3.4.8}$$

The above limitation values are aircraft specific and can be varying with the flight conditions. For the purpose of this study, the most conservative values of each limitation bound are used as a static limitation. Table B.1 summarized all the values used in this work.

**Aircraft Dynamics Model** By substituting the force models, in (3.4.3) and (3.4.4), into the equation of motion in (3.4.1), the standard state-space form of the aircraft dynamics model can be written as:

$$\dot{\boldsymbol{x}}_{AC} = \boldsymbol{f}_{AC}(\boldsymbol{x}_{AC}, \boldsymbol{u}_{AC}), \tag{3.4.9a}$$

$$\boldsymbol{x}_{AC} = [\ p_{\mathrm{H},N} \quad p_{\mathrm{H},E} \quad p_{\mathrm{H},D} \quad V_a \quad \chi_a \quad \gamma_a\ ]^T, \tag{3.4.9b}$$

$$\boldsymbol{u}_{AC} = [\ \bar{T} \quad n_z \quad \mu\ ]^T \tag{3.4.9c}$$

where $\boldsymbol{x}_{AC}$ is the aircraft state vector, $\boldsymbol{u}_{AC}$ is the aircraft control vector and $\boldsymbol{f}_{AC}(\boldsymbol{x}_{AC}, \boldsymbol{u}_{AC})$ is the resulting non-linear vector-value function of the aircraft dynamic model.

### 3.4.2 Navigation System

The navigation system model takes as input the true values of the aircraft state and then generate the necessary outputs for the purposes of *output tracking*, *state feedback*, and *navigation*. Therefore, the outputs are grouped into three sets, as will be seen in Figure 3.11.

Firstly, the **tracked output** vector, denoted by $\boldsymbol{y}$, contains the system variables required by the *outer tracking loop* of the manoeuvre autopilot. In order to track the airspeed, aircraft heading, and aerodynamic flight-path angle commands, the following output vector is required:

$$\boldsymbol{y} = [\ V_a \quad \chi_a \quad \gamma_a\ ]^T. \tag{3.4.10}$$

Secondly, the **feedback state** vector, denoted by $\boldsymbol{x}_{fbk}$, contains the system variables required by the *inner feedback-linearization* loop of the manoeuvre autopilot. In order to track the above commands, the following feedback state vector is required:

$$\boldsymbol{x}_{fbk} = [\ h \quad V_a \quad \chi_a \quad \gamma_a \quad \dot{V}_a \quad \dot{\chi}_a \quad \dot{\gamma}_a\ ]^T. \tag{3.4.11}$$

Thirdly, the **host's navigation state** vector, denoted by $\boldsymbol{x}_{nav}$, is obtained by imposing a random error $\epsilon$ on the actual values of the host state vector, that is:

$$\boldsymbol{x}_{nav} = \boldsymbol{x}_{\mathrm{H}} + \boldsymbol{\epsilon} \tag{3.4.12}$$

where the host state vector is defined as:

$$\boldsymbol{x}_{\mathrm{H}} = \begin{bmatrix} p_{\mathrm{H},N} & p_{\mathrm{H},E} & p_{\mathrm{H},D} & v_{\mathrm{H},N} & v_{\mathrm{H},E} & v_{\mathrm{H},D} & \phi & \theta & \psi \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{p}_{\mathrm{H}} \\ \boldsymbol{v}_{\mathrm{H}} \\ \boldsymbol{\Psi} \end{bmatrix} \tag{3.4.13}$$

and $\phi$, $\theta$, $\psi$ and $\boldsymbol{\Psi}$ denote the roll, pitch, yaw and aircraft attitude vector, respectively; and from the assumptions in Section 3.4, we have:

$$\phi = \mu, \qquad \theta = \alpha + \gamma_a, \qquad \psi = \chi_a. \tag{3.4.14}$$

### 3.4.3 Manoeuvre Autopilot

The design of the manoeuvre autopilot is based on the idea of NDI, which has been widely used in guidance and flight control system design, such as Snell *et al.* [1992] and Möckli [2006], and in aircraft performance simulation, such as Fisch [2011] and Hoffren & Sailaranta [2001]. Figure 3.11 shows the control scheme of NDI, which consists of two control loops: 1) an inner feedback linearization loop to transform the nonlinear system to a linear one; and 2) an outer output tracking loop to make the system outputs to track the given commands with the required performance. The transformation is achieved by making use of the known nonlinearity of the system, which is encapsulated in the Control Allocation block. The controller design for the transformed linear system, according to the given Tracking Performance, will be described in Desired Dynamic Controller.

**Control Allocation** The Control Allocation process deals directly with the known nonlinearities, i.e. the nonlinear aircraft dynamics and performance limitation (saturation) here. Although the Control Allocation technique is normally applied to over-actuated systems and involves solving an optimization problem, see Johansen & Fossen [2013], we used a virtual control vector of the same dimension as the tracked output and a simple algorithm to determine the aircraft controls for this work. Figure 3.12 shows this process with two steps: Dynamic Inversion and Constraint Handling.

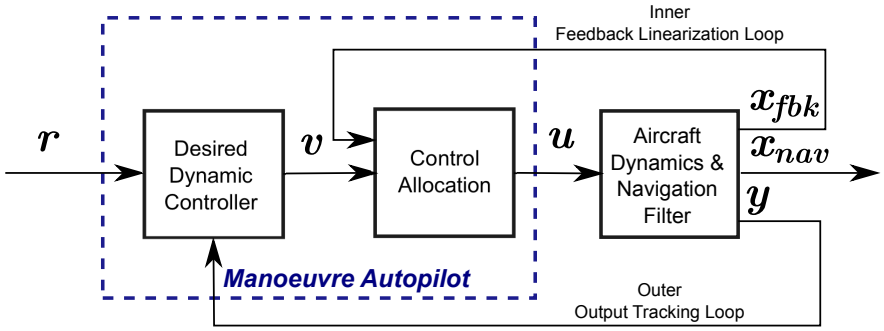



Figure 3.11: The control scheme for the manoeuvre autopilot, which is based on Nonlinear Dynamic Inversion technique.

**Dynamic Inversion** Firstly, as shown in the signal flow diagram in Figure 3.12, we select the virtual control vector $\boldsymbol{\nu}$[1] as:

$$\boldsymbol{\nu} = \begin{bmatrix} \dot{V}_a & \dot{\chi} & \dot{\gamma} \end{bmatrix}^T \tag{3.4.15}$$

so that the *virtual system* from $\boldsymbol{\nu}$ to $\boldsymbol{y}$ is a linear system with three poles at the origin. The current value of the virtual control will be commanded to the Dynamic Inversion process and served as the reference value for the first derivative of the tracked output:

$$\dot{\boldsymbol{y}}^{ref,DD}(t) = \boldsymbol{\nu}(t) \tag{3.4.16}$$

the superscript $(^{ref,DD})$ denotes that the underlying signal contains the reference values provided by the Desired Dynamic block.

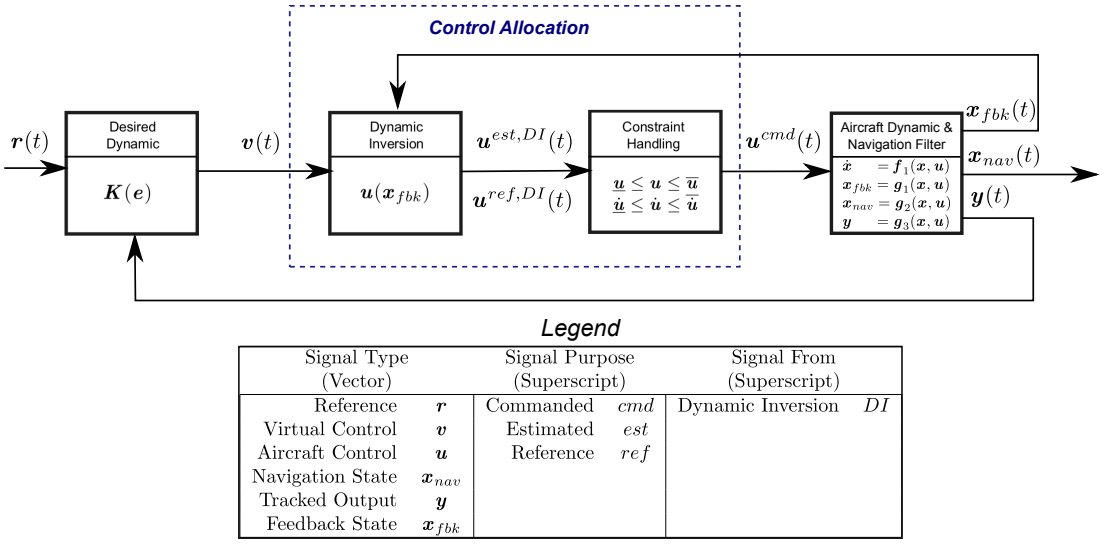
Then, the aircraft's equation of motion in (3.4.1) is rearranged to give the aircraft control as a function of the feedbacked state, i.e. $\boldsymbol{u}\left(\boldsymbol{x}_{fbk}\right)$:

$$\bar{T} = \frac{m\dot{V}_a + mg\sin\gamma + D}{T_{max}} \tag{3.4.17a}$$

$$\mu = \arctan\frac{V_a\dot{\chi}\cos\gamma}{V\dot{\gamma} + g\cos\gamma} \tag{3.4.17b}$$

$$n_z = \frac{g\cos\gamma + V_a\dot{\gamma}}{g\cos\mu} \tag{3.4.17c}$$

[1]Note that, $\boldsymbol{\nu}$ is a general vector, which is different from $\vec{\boldsymbol{v}}$ as a Euclidean vector.




*Legend*

| Signal Type (Vector) | | Signal Purpose (Superscript) | | Signal From (Superscript) | |
|---|---|---|---|---|---|
| Reference | $r$ | Commanded | $cmd$ | Dynamic Inversion | $DI$ |
| Virtual Control | $v$ | Estimated | $est$ | | |
| Aircraft Control | $u$ | Reference | $ref$ | | |
| Navigation State | $x_{nav}$ | | | | |
| Tracked Output | $y$ | | | | |
| Feedback State | $x_{fbk}$ | | | | |

Figure 3.12: The signal flow diagram for the Manoeuvre Autopilot.

where $D\left(V_a, h, n_z\right)$ and $T\left(V_a, h\right)$ can be calculated with (3.4.4) and (3.4.3).

Finally, using the function $\boldsymbol{u}\left(\boldsymbol{x}_{fbk}\right)$, the current values of the aircraft control vector can be estimated by the Dynamic Inversion block with:

$$\boldsymbol{u}^{est,DI}(t) = \boldsymbol{u}\left(\boldsymbol{x}_{fbk}(t)\right) \tag{3.4.18}$$

where $\boldsymbol{x}_{fbk}(t)$ is from the navigation system.

The reference values of the aircraft control vector are generated by the Dynamic Inversion block with:

$$\boldsymbol{u}^{ref,DI}(t) = \boldsymbol{u}\left(\boldsymbol{x}_{fbk}^{est,DI}(t)\right) \tag{3.4.19}$$

where $\boldsymbol{x}_{fbk}^{est,DI}(t)$ is the estimated value of the state feedback vector, internally made by the Dynamic Inversion block using:

$$\boldsymbol{x}_{fbk}^{est,DI}(t) = \left[h(t) \quad V_a(t) \quad \chi_a(t) \quad \gamma_a(t) \quad \dot{V}_a{}^{ref,DD}(t) \quad \dot{\chi}_a{}^{ref,DD}(t) \quad \dot{\gamma}_a{}^{ref,DD}(t)\right]^T \tag{3.4.20}$$

**Constraint Handling** With the estimated and reference values of the aircraft control vector, as given in (3.4.18) and (3.4.19), and their limitation values given by (3.4.8),

the Constraint Handling Algorithm given in Algorithm 3.1 is used to ensure that the commanded aircraft control vector is attainable:

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}^{cmd} \leq \boldsymbol{u}_{min}, \quad \dot{\boldsymbol{u}}_{min} \leq \dot{\boldsymbol{u}}^{cmd} \leq \dot{\boldsymbol{u}}_{min} \tag{3.4.21}$$

Besides imposing the rate and range limit on the control signal, the algorithm makes sure that the constraints are imposed in the correct order, so as to take into account the dependency between the $n_z$ and $\mu$, as shown in (3.4.17).

**Algorithm 3.1:** Constraint Handling Algorithm

**Input**: $\boldsymbol{u}^{ref,DI}, \boldsymbol{u}^{est,DI}, \boldsymbol{u}_{max}, \boldsymbol{u}_{min}, \dot{\boldsymbol{u}}_{max}, \dot{\boldsymbol{u}}_{min}$
**Output**: $\boldsymbol{u}^{cmd}$

$\Delta t \leftarrow$ Sample time;
**for** $\forall u_i^{ref,DI} \in \{\bar{T}, \mu, n_z\}$ **do**
  $\dot{u}_i^{ref,DI} \leftarrow \dfrac{u_i^{ref,DI} - u_i^{est,DI}}{\Delta t}$ ;
  **if** $\dot{u}_i^{ref,DI} > \dot{u}_{i,max}$ **then**
    $u_i^{cmd} = u_i^{est,DI} + u_{i,max}\Delta t$;
  **else if** $\dot{u}_i^{ref,DI} < \dot{u}_{i,min}$ **then**
    $u_i^{cmd} = u_i^{est,DI} - u_{i,min}\Delta t$;
  **else**
    $u_i^{cmd} = u_i^{ref,DI}$;
  **end**
  **if** $u_i^{cmd} > u_{i,max}$ **then**
    $u_i^{cmd} = u_{i,max}$;
  **else if** $u_i^{cmd} < u_{i,min}$ **then**
    $u_i^{cmd} = u_{i,min}$;
  **end**
**end**

**Desired Dynamics Controller** Figure 3.13 shows the signal flow graph for the desired dynamics controller, in which the transformed (virtual) system, assuming perfect dynamic inversion, appears to be a linear system with three parallel and uncoupled integrators. The task is now reduced to designing a linear controller to drive the error signal to zero.

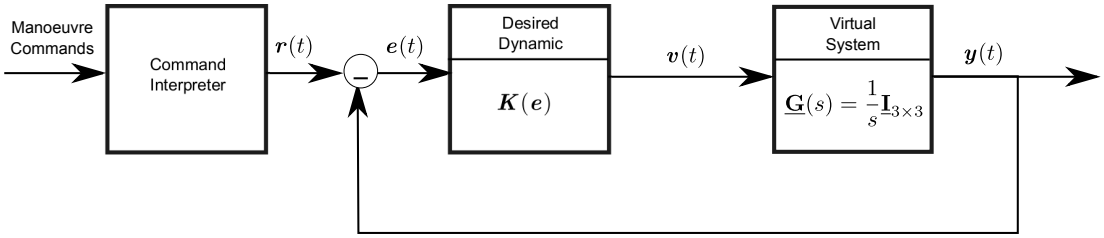



Figure 3.13: The Block Diagram for the Desired Dynamics Controller.

The linear controller can be designed (or simply tuned) to match the given tracking and transient performance of the platform being tested. This work uses a simple proportional controller of the form $\boldsymbol{K} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ :

$$\boldsymbol{v} = \boldsymbol{K}(\boldsymbol{e}) = \boldsymbol{K}(\boldsymbol{r} - \boldsymbol{y}) \tag{3.4.22}$$

The signal flow is:

$$\boldsymbol{v}(t) = \underline{\mathbf{K}}(\boldsymbol{r}(t) - \boldsymbol{y}(t)) \tag{3.4.23}$$

with:

$$\boldsymbol{v}(t) = \begin{bmatrix} \dot{V}_a^{ref,DD}(t) \\ \dot{\chi}_a^{ref,DD}(t) \\ \dot{\gamma}_a^{ref,DD}(t) \end{bmatrix}, \quad \boldsymbol{r}(t) = \begin{bmatrix} V_a^{ref}(t) \\ \chi_a^{ref}(t) \\ \gamma_a^{ref}(t) \end{bmatrix}, \quad \boldsymbol{y}(t) = \begin{bmatrix} V_a(t) \\ \chi_a(t) \\ \gamma_a(t) \end{bmatrix}$$

$$\underline{\mathbf{K}} = \begin{bmatrix} K_{V_a} & 0 & 0 \\ 0 & K_{\chi_a} & 0 \\ 0 & 0 & K_{\gamma_a} \end{bmatrix}$$

where:

$K_{\otimes}$: is the gain of the controller, which can be regarded as the reciprocal of the time constant $\tau_{\otimes}$ of the corresponding control channel $\otimes$.

Lastly, as shown in the Command Interpreter block in Figure 3.13, some operation may be required to transform the external manoeuvre commands (from the collision avoidance system or encounter model) to the reference values of the reference vector

$\boldsymbol{r}$. For example, the vertical speed commands $V_v^{cmd}$ can be transformed to the reference value of aerodynamic flight-path angle as, assuming there is no vertical wind:

$$\gamma^{ref}(t) = \arcsin \frac{V_v^{cmd}(t)}{V_a(t)} \tag{3.4.24}$$

## 3.5 Surveillance System Model

In order to be able to study the effect of the FOR restriction—i.e. the effects caused by the restricted *Field of View* and maximum detection range, referred to as FOR effect hereafter, the following surveillance system conditions are required:

1. **Ideal** condition, serving as a theoretical limit, which uses the true intruder state for collision avoidance.

2. **Noisy** condition, emulating the cooperative situation, which uses the noisy intruder state subject to random error.

3. **Limited FOR** condition, serving as a theoretical basis to investigate the FOR effect, which uses the FOR-restricted intruder state.

4. **Realistic** condition, emulating the non-cooperative situation, which uses the FOR-restricted, noisy intruder state.

Figure 3.14 shows the architecture of the surveillance system model that is developed to provide the above conditions. The model takes as input the true states of the host and intruder ($\boldsymbol{x}_{\mathrm{H}}$ and $\boldsymbol{x}_{\mathrm{I}}$) and generates a tracker state vector $\boldsymbol{x}_{tra}$, via the following intermediate processes:

1. calculates the actual relative position between the two aircraft;

2. converts the relative position to the relative range, azimuth and elevation and only outputs them when they are within the FOR limits, so as to emulate the FOR-restricted measurements;

3. maintains the intruder track status according to the FOR-restricted measurements;

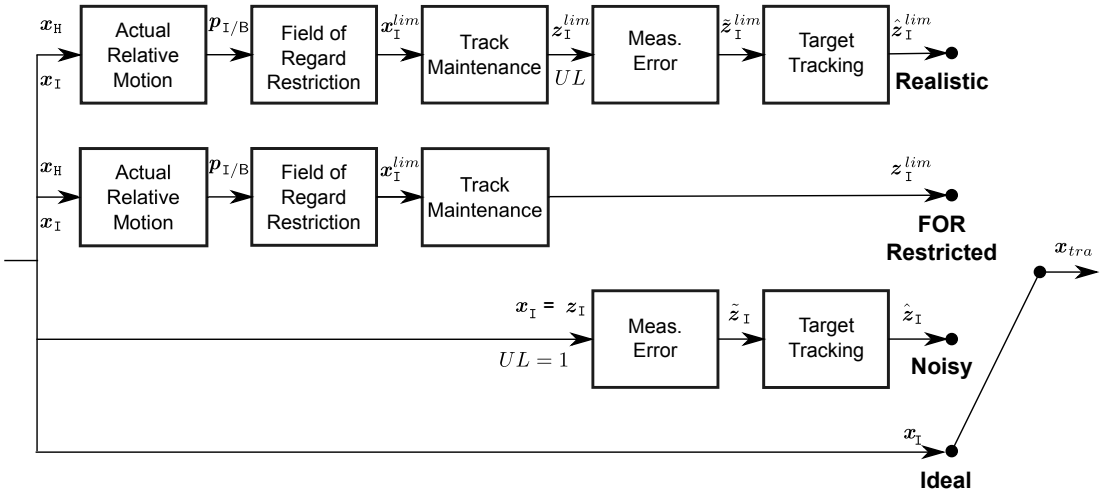



Figure 3.14: Architecture of surveillance system model, showing the generation of four types of test cases: 1. *Ideal*; 2. *Noisy*; 3. *FOR restricted*; 4. *Realistic*.

4. adds random errors to the measurements according to the current track status; and

5. filters the noisy measurements in order to emulate the behaviour of the actual tracker.

### 3.5.1 Actual Relative Motion

Figure 3.15 shows the relative geometry between the host and intruder. Three frames of reference are required to calculate the relative motion and the FOR-restricted measurements:

1. Navigation (inertial) frame $\mathcal{N}$: a frame of reference, non-rotating and non-translating with respect to the rigid Earth. The frame has its origin N at an arbitrary position (defined for each simulation) on the Earth's surface, and its base triad oriented in the north, east, and down directions. This is regarded as an inertial frame.

2. Host-carried frame $\mathcal{H}$: a frame of reference, translating with the rigid host aircraft. The frame has its origin H at the Centre of Mass (CM) of the host aircraft, and its base triad oriented in the north, east, and down directions.

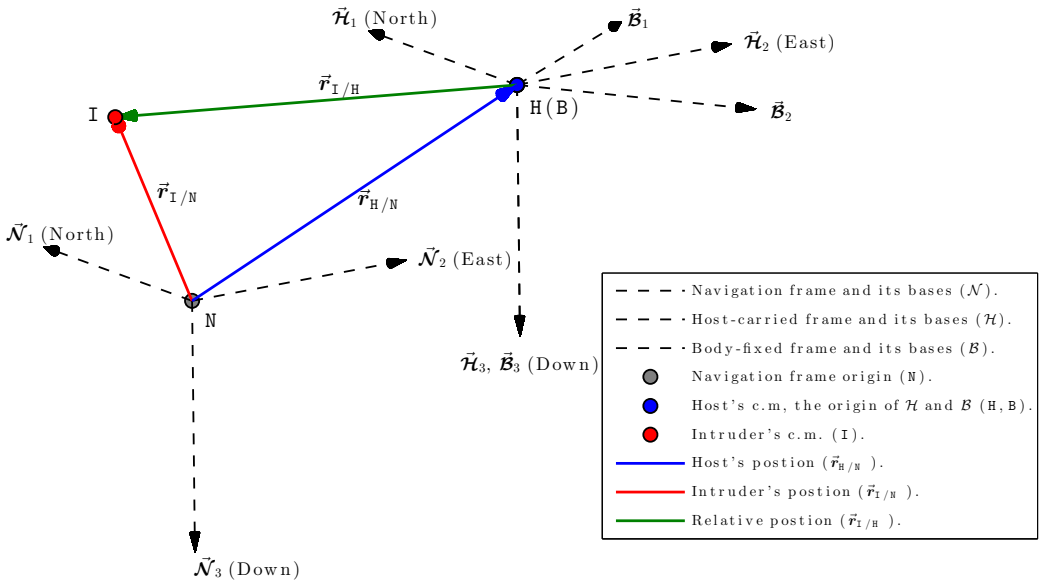



Figure 3.15: Relative motion geometry, illustrating 1) the relative position between the host and intruder and 2) thee frames of reference: navigation, host and body-fixed frames.

3. Body-fixed frame $\mathcal{B}$: a frame of reference, translating with the rigid host aircraft. The frame has its origin B at the CM of the host aircraft, and its first base vector parallel to the fuselage reference line, and its third base vector in the aircraft plane of symmetry.

The relative motion is described by the displacement vector $\vec{\boldsymbol{r}}_{\mathtt{I/H}}$ between the CM of the intruder I and host H. The figure also illustrates the following vector equations:

$$\vec{\boldsymbol{r}}_{\mathtt{I/H}} = \vec{\boldsymbol{r}}_{\mathtt{I/N}} - \vec{\boldsymbol{r}}_{\mathtt{H/N}} \tag{3.5.1}$$

$$= \vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{N}} - \vec{\boldsymbol{p}}_{\mathtt{H}/\mathcal{N}} \quad (\because \mathtt{N} \text{ is the base origin of } \mathcal{N}) \tag{3.5.2}$$

$$\iff \vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{H}} = \vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{N}} - \vec{\boldsymbol{p}}_{\mathtt{H}/\mathcal{N}} \quad (\because \mathtt{H} \text{ is the base origin of } \mathcal{H}) \tag{3.5.3}$$

$$= \vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{B}} \quad (\because \mathcal{H} \text{ and } \mathcal{B} \text{ share the same base origin H(B)}) \tag{3.5.4}$$

$\vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{H}}$ and $\vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{B}}$ are, respectively, the position vectors of the intruder in the host and body frame and both position vectors are equal to the displacement vector $\vec{\boldsymbol{r}}_{\mathtt{I/H}}$.which

is referred to as the relative position vector of the intruder with respect to the host.

Expressing the relative position (3.5.3) in the Host coordinate system $\mathsf{H}$ yields:

$$\begin{aligned}\left[\vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{H}}\right]^{\mathsf{H}} &= \left[\vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{N}}\right]^{\mathsf{H}} - \left[\vec{\boldsymbol{p}}_{\mathtt{H}/\mathcal{N}}\right]^{\mathsf{H}} \\ &= \underbrace{\underline{\mathbf{C}}_{\mathsf{N}}^{\mathsf{H}}}_{\underline{\mathbf{I}}_{3\times 3}} \left( \underbrace{\left[\vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{N}}\right]^{\mathsf{N}}}_{\equiv \boldsymbol{p}_{\mathtt{I}}} - \underbrace{\left[\vec{\boldsymbol{p}}_{\mathtt{H}/\mathcal{N}}\right]^{\mathsf{N}}}_{\equiv \boldsymbol{p}_{\mathtt{H}}} \right) \end{aligned} \tag{3.5.5}$$

$$\therefore \quad \boldsymbol{p}_{\mathtt{I}/\mathsf{H}} \equiv \left[\vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{H}}\right]^{\mathsf{H}} = \boldsymbol{p}_{\mathtt{I}} - \boldsymbol{p}_{\mathtt{H}}, \tag{3.5.6}$$

where $\underline{\mathbf{C}}_{\mathsf{N}}^{\mathsf{H}}$[1] is the direction cosine matrix between the host and navigation coordinate system; $\boldsymbol{p}_{\mathtt{H}}$ and $\boldsymbol{p}_{\mathtt{I}}$ are the host and intruder (absolute) position vector.

Similarly, the relative position can also be expressed in the Body coordinate system with the following equations:

$$\begin{aligned}\left[\vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{B}}\right]^{\mathsf{H}} &= \left[\vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{H}}\right]^{\mathsf{H}} && \text{(from (3.5.4))} \\ &= \boldsymbol{p}_{\mathtt{I}} - \boldsymbol{p}_{\mathtt{H}} && \text{(from (3.5.6))}\end{aligned} \tag{3.5.7}$$

$$\iff \underline{\mathbf{C}}_{\mathsf{H}}^{\mathsf{B}} \left[\vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{B}}\right]^{\mathsf{H}} = \underline{\mathbf{C}}_{\mathsf{H}}^{\mathsf{B}} \left(\boldsymbol{p}_{\mathtt{I}} - \boldsymbol{p}_{\mathtt{H}}\right) \tag{3.5.8}$$

$$\therefore \quad \boldsymbol{p}_{\mathtt{I}/\mathsf{B}} \equiv \left[\vec{\boldsymbol{p}}_{\mathtt{I}/\mathcal{B}}\right]^{\mathsf{B}} = \underline{\mathbf{C}}_{\mathsf{H}}^{\mathsf{B}} \left(\boldsymbol{p}_{\mathtt{I}} - \boldsymbol{p}_{\mathtt{H}}\right) \tag{3.5.9}$$

where $\underline{\mathbf{C}}_{\mathsf{H}}^{\mathsf{B}}$ is the direction cosine matrix from the host to body-fixed coordinate system, which is a function of the aircraft attitude, i.e. $\underline{\mathbf{C}}_{\mathsf{H}}^{\mathsf{B}}(\boldsymbol{\Psi})$.

In summary, expressing the relative position vector $\vec{\boldsymbol{r}}_{\mathtt{I}/\mathsf{H}}$ in the host and body-fixed coordinate system ($\mathsf{H}$ and $\mathsf{B}$) yields two *different* coordinate vectors ($\boldsymbol{p}_{\mathtt{I}/\mathsf{H}}$ and $\boldsymbol{p}_{\mathtt{I}/\mathsf{B}}$). Both of these are also functions of the host and intruder state vector ($\boldsymbol{x}_{\mathtt{H}}$ and $\boldsymbol{x}_{\mathtt{I}}$), i.e. $\boldsymbol{p}_{\mathtt{I}/\mathsf{H}}(\boldsymbol{x}_{\mathtt{H}},\boldsymbol{x}_{\mathtt{I}})$ and $\boldsymbol{p}_{\mathtt{I}/\mathsf{B}}(\boldsymbol{x}_{\mathtt{H}},\boldsymbol{x}_{\mathtt{I}})$.

### 3.5.2 Field of Regard

A multi-sensor configuration, combing a set of electro-optical/infrared (EO/IR) sensors and an on-board radar, is regarded as a promising solution for non-cooperative sensing,

[1] As there is no rotation between the host and navigation frame (so as their coordinate systems, and therefore $\underline{\mathbf{C}}_{\mathsf{N}}^{\mathsf{H}} = \mathbf{I}_{3\times 3}$), the two components vectors are the same when an Euclidean vector is expressed in these two coordinate systems, i.e. $[\vec{\boldsymbol{a}}]^{\mathsf{H}} = [\vec{\boldsymbol{a}}]^{\mathsf{N}}$.

Table 3.4: An example set of surveillance sensor parameters.

| Parameter | Symbol | Unit | Radar | EO/IR |
|---|---|---|---|---|
| Relative range limitation | $r_{max}$ | NM | 5 | 5 |
| Azimuth limitation | $\Phi_{lim}$ | deg | ±110 | ±110 |
| Elevation limitation | $\Theta_{lim}$ | deg | ±15 | ±15 |
| Relative range error standard deviation | $\sigma_r$ | ft | 50 | N/A |
| Relative range rate error standard deviation | $\sigma_{\dot{r}}$ | ft/s | 10 | N/A |
| Bearing error standard deviation | $\sigma_\Phi$ | deg | 1 | 0.5 |
| Bearing rate error standard deviation | $\sigma_{\dot{\Phi}}$ | deg/s | N/A | 0.5 |
| Elevation error standard deviation | $\sigma_\Theta$ | deg | 1 | 0.5 |
| Elevation rate error standard deviation | $\sigma_{\dot{\Theta}}$ | deg/s | N/A | 0.5 |

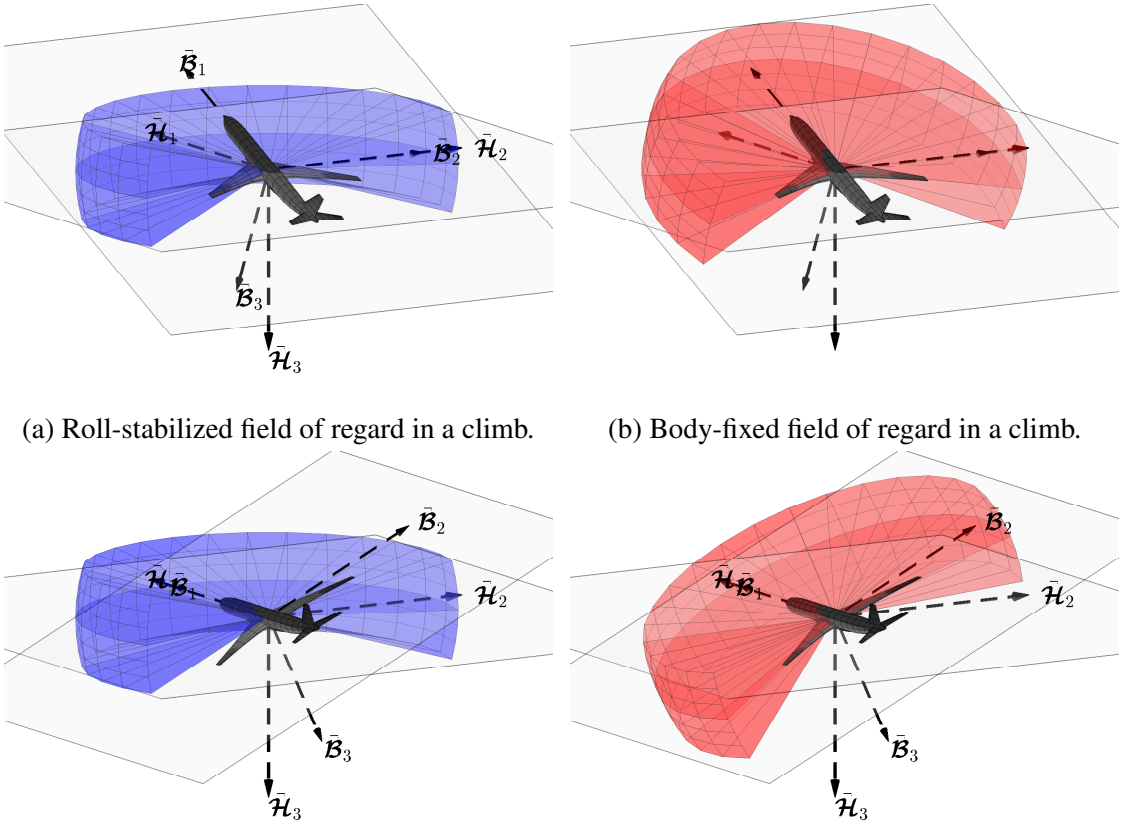
see Fasano *et al.* [2008]; Chen *et al.* [2009]; Patchett & Ansell [2010].

**Sensor Performance** Table 3.4 summarizes an example set of performance parameters for this sensor suite, based on Temizer *et al.* [2010] and F38 Committee [2007]. The sensor parameters are given in a spherical coordinate system, i.e. the limits and error standard deviation on the relative range, azimuth, and elevation.

**Field of Regard** According to the specified limits, Figure 3.16 shows the sensor suite's field of regard, within which the sensors make measurements. As can be seen from its shape, the field of regard is determined by the range (relative range limit) and field of view (azimuth and elevation limits). Depending on the system configurations, the field of view can be either body-fixed or roll-stabilized.

**Body-fixed Measurement** Figures 3.16b and 3.16d show the body-fixed field of regard, which can be conveniently expressed in the body-fixed spherical coordinate system. A target measurement is given by its relative position's spherical coordinates, i.e. the relative range $r_{\mathsf{B_s}}$, azimuth $\Phi_{\mathsf{B_s}}$ and elevation $\Theta_{\mathsf{B_s}}$:

$$\boldsymbol{p}_{\mathtt{I}/\mathsf{B_s}} = \begin{bmatrix} r_{\mathsf{B_s}} & \Phi_{\mathsf{B_s}} & \Theta_{\mathsf{B_s}} \end{bmatrix}^T, \tag{3.5.10}$$




(a) Roll-stabilized field of regard in a climb.

(b) Body-fixed field of regard in a climb.

(c) Roll-stabilized field of regard in a turn.

(d) Body-fixed field of regard in a turn.

Figure 3.16: Sensor suite's field of regard.

where the subscript $(\cdot)_{\mathtt{B_s}}$ denotes that the variables are related to the body-fixed spherical coordinate system.

**Limited-Field-of-Regard Measurement** The relative position vector's spherical coordinates can be obtained from its rectangular coordinates:

$$\boldsymbol{p}_{\mathtt{I/B_s}} = \boldsymbol{T}_{\mathtt{r}}^{\mathtt{s}}\left(\boldsymbol{p}_{\mathtt{I/B}}\right). \tag{3.5.11}$$

with a coordinate transformation function:

$$\boldsymbol{T}_{\mathtt{r}}^{\mathtt{s}}\left(\boldsymbol{p}_{\mathtt{I/B}}\right) = \begin{bmatrix} \sqrt{x^2+y^2+z^2} \\ \arctan(\frac{y}{x}) \\ -\arcsin(\frac{z}{r}) \end{bmatrix}.$$

Only the coordinate values that fall within the field of regard limits, as given by Table 3.4, will be outputted to simulate the FOR-restricted measurements:

$$\boldsymbol{z}_{\mathtt{I}}^{lim} = \boldsymbol{x}_{\mathtt{I}}^{lim} = \begin{cases} \boldsymbol{x}_{\mathtt{I}} & \text{if } \left(r_{\mathtt{B_s}} \leq r_{max}\right) \wedge \left(\left|\Phi_{\mathtt{B_s}}\right| \leq \Phi_{lim}\right) \wedge \left|\Theta_{\mathtt{B_s}} \leq \Theta_{lim}\right| \\ \boldsymbol{0}_{6\times 1} & \text{else} \end{cases} \tag{3.5.12}$$

### 3.5.3 Track Maintenance

According to Zeitlin [2012], a tracker should be able to:

1. initiate a target track whilst gaining sufficient confidence that the target detection is valid;
2. update the target track whilst gaining sufficient confidence that the measurements are valid;
3. maintain the target track even in the absence of valid measurements; and
4. drop the target track according to some design criteria.

Figure 3.17 shows a state machine representing the high-level view of the tracker model, in which four states are used to represent the tracker's mode of operations: `No_Track`, `New_Track`, `Updated_Track`, and `Predicted_Track`.

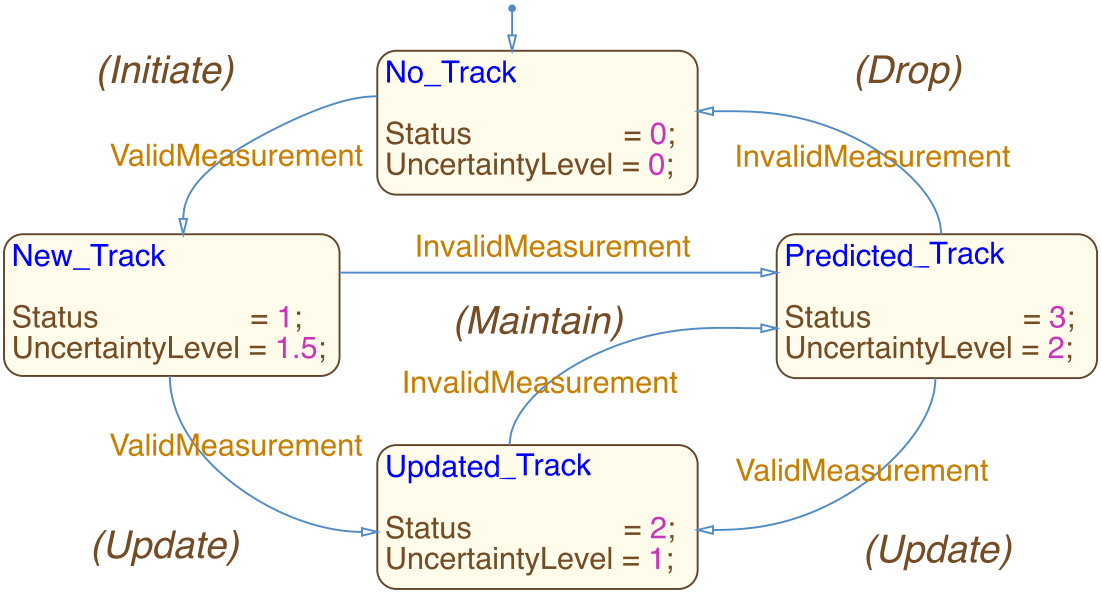



Figure 3.17: The finite state machine used to model the status of a target track.

The transitions between the states are guarded by two conditions: `ValidMeasurement` and `InvalidMeasurement`. For the model in this work, `ValidMeasurement` will be satisfied, if 80% of the previous measurements, over the update interval of the tracker system[1], are valid; otherwise the other condition `InvalidMeasurement` will be satisfied.

According to the active state, a series of actions (such as track prediction, measurement comparison, and track update) will be triggered. It is assumed in this work that the final product after all those actions is simply an estimate of the target's position and velocity, subject to different level of uncertainty. Therefore, depending on the active mode, this block will output the corresponding uncertainty level (denoted by $UL$) along with the limited-field-of-regard measurement ($z_{\mathrm{I}}^{lim}$), which will be used in the measurement error model to emulate the uncertainty in the tracker system.

[1]Note that, the update intervals of the sensors and of the tracker are normally different. The tracker's update interval would depend on the technology, and typically would lie within 1 to 5 second, [Zeitlin, 2012, § 2.7], while the update interval for the sensor can be as soon as 0.1 second [Fasano, 2008, § 2.2].

### 3.5.4 Measurement Error

Gaussian noise is added to the measurement vector to emulate the noisy measurements:

$$\tilde{z}_{\mathtt{I}} = z_{\mathtt{I}} + \kappa_{tra} UL\boldsymbol{\epsilon} \quad \text{or} \quad \tilde{z}_{\mathtt{I}}{}^{lim} = z_{\mathtt{I}}^{lim} + \kappa_{tra} UL\boldsymbol{\epsilon}, \tag{3.5.13}$$

where $\tilde{(\cdot)}$ denotes the noisy values of the underlying variable, $\kappa_{tra}$ is a simulation parameter used to scale up the random error to emulate the desired level of tracking uncertainty, $UL \in \{0, 1, 1.5, 2\}$ is the uncertainty level corresponding to the current track status; $\boldsymbol{\epsilon}$ is the random error vector:

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{p_N} & \epsilon_{p_E} & \epsilon_{p_D} & \epsilon_{v_N} & \epsilon_{v_E} & \epsilon_{v_D} \end{bmatrix}^T \tag{3.5.14}$$

with

$$\epsilon_i \sim \mathcal{N}(0, \sigma_i^2), \quad \forall i \in \{p_N, p_E, p_D, v_N, v_E, v_D\}$$

where, $\mathcal{N}$ is the normal distribution, $\sigma$ is standard deviation as given by Table 3.5.

Table 3.5: Measurement error model parameters.

| Standard Deviation of | Symbol | Value | Unit |
|---|---|---|---|
| North position | $\sigma_{p_N}$ | 50 | ft |
| East position | $\sigma_{p_E}$ | 50 | ft |
| Down position | $\sigma_{p_D}$ | 50 | ft |
| North speed | $\sigma_{v_N}$ | 10 | ft/s |
| East speed | $\sigma_{v_E}$ | 10 | ft/s |
| Down speed | $\sigma_{v_D}$ | 10 | ft/s |

[1] It is assumed that the error magnitudes, of the position and speed, are similar to those of the relative range and range rate of a radar, as specified in the Table 3.4.

### 3.5.5 Target Tracking

With the noisy measurement vector, an Alpha-Beta-Gamma Filter is used to estimate the intruder's position and velocity.

**Initialization Process** Upon receiving the first valid measurement $\tilde{z}_{\mathrm{I}}$, the filter initializes the estimate $\hat{z}_{\mathrm{I}}$ and its derivatives using:

$$z^{est}(0) = z^{mes}(0) \tag{3.5.15a}$$
$$\dot{z}^{est}(0) = \mathbf{0}_{6\times 1} \tag{3.5.15b}$$
$$\ddot{z}^{est}(0) = \mathbf{0}_{6\times 1} \tag{3.5.15c}$$

with

$$z^{mes} = \tilde{z}_{\mathrm{I}} \quad \text{and} \quad \hat{z}_{\mathrm{I}} = z^{est} \tag{3.5.16}$$

where $z^{est}$, $\dot{z}^{est}$, and $\ddot{z}^{est}$ are the internal states of the filter and will be updated subsequently with the following two steps.

**The prediction step**

$$z^{pre}(k) = z^{est}(k-1) + \dot{z}^{est}(k-1)\Delta t + \frac{1}{2}\ddot{z}^{est}(k-1)\Delta t^2 \tag{3.5.17a}$$
$$\dot{z}^{pre}(k) = \dot{z}^{est}(k-1) + \ddot{z}^{est}(k-1)\Delta t \tag{3.5.17b}$$
$$\ddot{z}^{pre}(k) = \ddot{z}^{est}(k-1) \tag{3.5.17c}$$

**The correction (update) step**

$$z^{est}(k) = z^{pre}(k) + \underline{\alpha}\left(z^{mes}(k) - z^{pre}(k)\right) \tag{3.5.18a}$$
$$\dot{z}^{est}(k) = \dot{z}^{pre}(k) + \underline{\beta}\left[z^{mes}(k) - z^{pre}(k)\right]\frac{1}{\Delta t} \tag{3.5.18b}$$
$$\ddot{z}^{est}(k) = \ddot{z}^{pre}(k) + \underline{\gamma}\left[z^{mes}(k) - z^{pre}(k)\right]\frac{1}{\Delta t^2} \tag{3.5.18c}$$
$$k = 1, \ldots, N$$

where, $k$ is the time step, $N$ is the time step when there is an invalid measurement, $\Delta t$ is the update interval, and $\underline{\alpha}$, $\underline{\beta}$, and $\underline{\gamma}$ are three diagonal matrices with the filter gains on the diagonals.

The filter gains determine the relative degree of reliance on current and previous measurements. The larger the gains, the more confidence placed on the measurement;

gains of unity would place complete reliance on the current measurement and result in no filtering. For this work, the following gains are used:

$$\forall i = j: \quad \alpha_{i,j} = 0.4, \quad \beta_{i,j}, = 0.1 \quad \gamma_{i,j} = 0.01 \tag{3.5.19}$$

which are the minimum values of the gains for the relative range, range rate, and range acceleration in an adaptive alpha-beta-gamma tracker used in ICAO [2007].

## 3.6 Collision Avoidance Logic Model

To serve as a baseline, this section describes a collision avoidance logic based on a geometric CDR approach developed by Fasano *et al.* [2008]; Carbone *et al.* [2006]; Luongo *et al.* [2009].

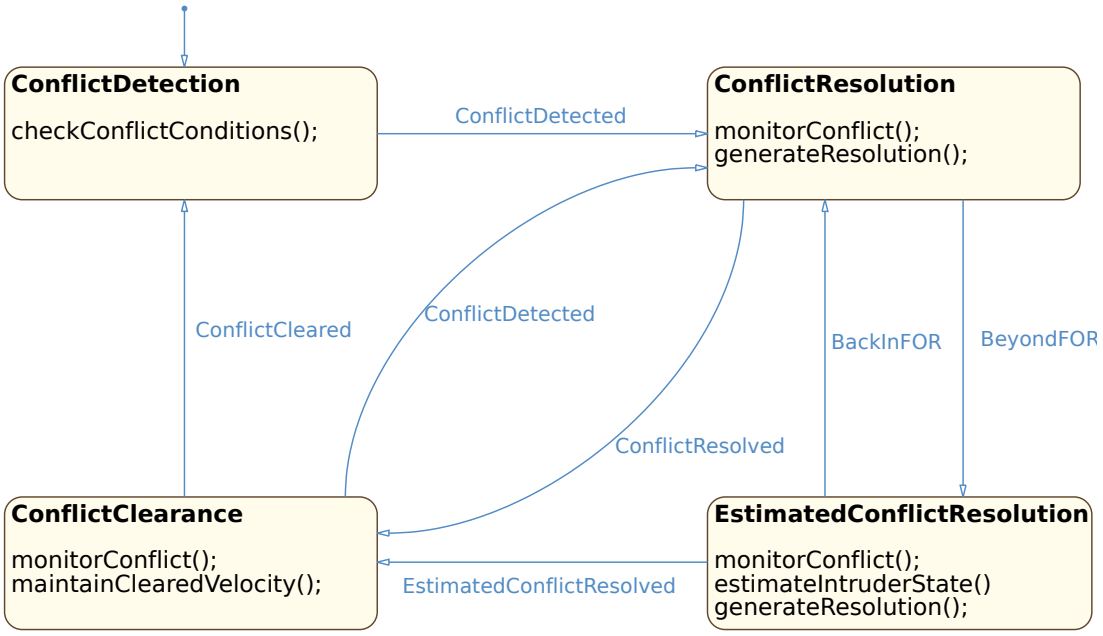



Figure 3.18: State machine for the collision avoidance logic, constructed according to Fasano *et al.* [2008].

Figure 3.18 shows the state machine of the collision avoidance logic, which takes as input the navigation states $\boldsymbol{x}_{nav}$ and tracker states $\boldsymbol{x}_{tra}$ and outputs the host manoeuvre

commands $\boldsymbol{c}_{\mathrm{H}}$. The logic works in the following four modes:

- in its initial `ConflictDetection` state, the logic continuously checks the conflict conditions as given in § 3.6.1;
- if the conflict conditions are met, the logic will trigger a transition to the state `ConflictResolution`, in which the resolution manoeuvre command will be generated (as described in §3.6.2) and the detected conflict will be monitored thereafter (as explained in §3.6.3);
- once the conflict is resolved, a transition to the `ConflictClearance` state will be triggered, and the last cleared velocity will be outputted as the manoeuvre command until the the conflict is cleared, or unless the conflict conditions are satisfied again;
- the `EstimatedConflictResolution` state is included to handle the situation when the intruder goes beyond the FOR. In this situation the intruder state will be estimated by linearly projecting its last known state.

The specific definitions for the above triggering events can be found in the corresponding subsections.

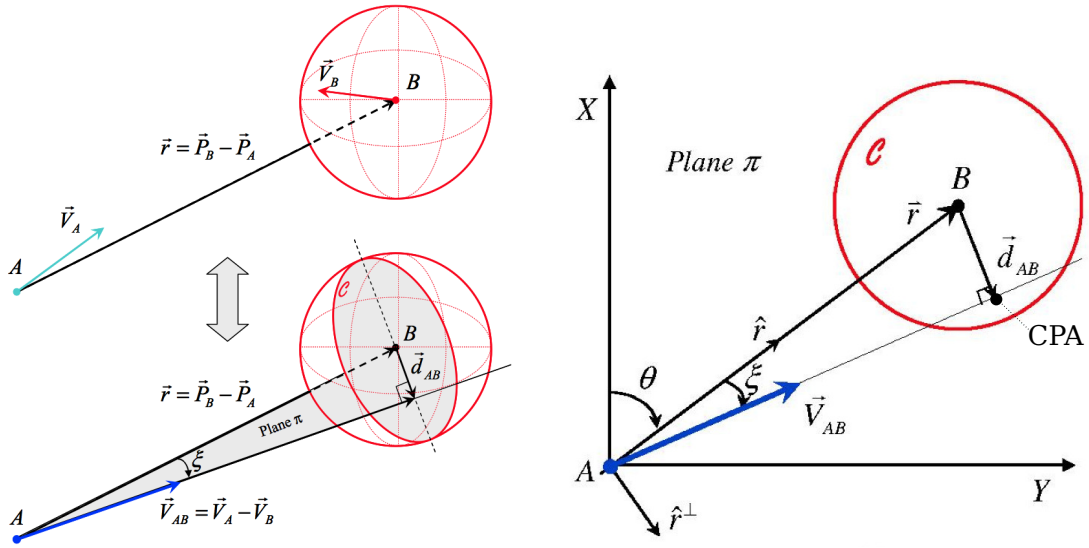
### 3.6.1 Conflict Detection

The conflict detection problem is to detect whether the displacement between the host and intruder is potentially, or even actually, less than the required separation minimum. We formulate the problem in the navigation coordinate system $\mathsf{N}$ (see Figure 3.7) and use the notation and solution given by Carbone *et al.* [2006].

**Relative Motion** Figure 3.19a shows the geometry of the problem, in which the intruder is denoted by the sphere $B$ with radius $R$ (referred to as Safety Bubble hereafter) and the host is denoted by the point $A$, that is:

$$\vec{\boldsymbol{P}}_A = [\hat{p}_{\mathrm{H},N} \quad \hat{p}_{\mathrm{H},E} \quad \hat{p}_{\mathrm{H},D}]^T \qquad \text{and} \qquad \vec{\boldsymbol{V}}_A = [\hat{v}_{\mathrm{H},N} \quad \hat{v}_{\mathrm{H},E} \quad \hat{v}_{\mathrm{H},D}]^T \tag{3.6.1}$$

$$\vec{\boldsymbol{P}}_B = [\hat{p}_{\mathrm{I},N} \quad \hat{p}_{\mathrm{I},E} \quad \hat{p}_{\mathrm{I},D}]^T \qquad \text{and} \qquad \vec{\boldsymbol{V}}_B = [\hat{v}_{\mathrm{I},N} \quad \hat{v}_{\mathrm{I},E} \quad \hat{v}_{\mathrm{I},D}]^T \tag{3.6.2}$$




(a) Relative motion geometry in the conflict detection problem.

(b) Collision plane $\boldsymbol{\pi}$ containing the vectors of the relative position $\vec{\boldsymbol{r}}_{\mathtt{A/B}}$ and velocity $\vec{\boldsymbol{V}}_{AB}$.

Figure 3.19: Illustration of conflict detection solution, adapted from Carbone *et al.* [2006].

where the estimated values are from the navigation ($\boldsymbol{x}_{nav}$) and tracker ($\boldsymbol{x}_{tra}$) system.

By introducing the relative position $\vec{\boldsymbol{r}}_{\mathtt{A/B}}$ and velocity $\vec{\boldsymbol{V}}_{AB}$ between the point $A$ and sphere $B$:

$$\vec{\boldsymbol{r}}_{\mathtt{A/B}} = \vec{\boldsymbol{P}}_B - \vec{\boldsymbol{P}}_A \quad \text{and} \quad \vec{\boldsymbol{V}}_{AB} = \vec{\boldsymbol{V}}_A - \vec{\boldsymbol{V}}_B,^{1} \tag{3.6.3}$$

and assuming that the velocities of the host and intruder remain constant (referred to as linear assumption hereafter), the relative motion can be regarded as: the point $A$ is moving relative to the sphere $B$ with the relative velocity $\vec{\boldsymbol{V}}_{AB}$.

**Closest Point of Approach** As can be seen from Figure 3.19b, if $A$ and $B$ are converging, the point $A$ will move along the direction of $\vec{\boldsymbol{V}}_{AB}$ and reach, after a certain time, the Closest Point of Approach (CPA) where the separation distance between $A$ and $B$ is minimum.[2] This point is referred to as *linear* CPA and denoted by $\mathtt{C}_{lin}$ here-

[1]Note that, $\vec{\boldsymbol{r}}$ is the relative position vector of point $B$ with respect to point $A$, while $\vec{\boldsymbol{V}}_{AB}$ is the realative veclocity vector of point A with respect to point B.

[2]Otherwise, if they are diverging (moving away from each other), the current position of the host will be the CPA.

after, where the'linear' is used to highlight that the variable is derived under the linear assumption.

**Linear Time-to-CPA** The relative range $r$ is defined as the Euclidean norm of the relative position vector; and the relative range rate $\dot{r}$ can be calculated by projecting the relative velocity $\vec{V}_{AB}$ onto $\vec{r}_{\mathrm{A/B}}$:

$$r = \|\vec{r}_{\mathrm{A/B}}\| \quad \text{and} \quad \dot{r} = -\frac{\vec{V}_{AB} \cdot \vec{r}_{\mathrm{A/B}}}{\|\vec{r}_{\mathrm{A/B}}\|} \tag{3.6.4}$$

Under the linear assumption, the linear time-to-CPA can be calculated with:

$$t_{\mathrm{C}_{lin}} = -\frac{r}{\dot{r}} \tag{3.6.5}$$

Alternatively, see Figure 3.19b, one can project $\vec{r}_{\mathrm{A/B}}$ onto $\vec{V}_{AB}$ and divide the resulting scalar projection by $\|\vec{V}_{AB}\|$ (the relative speed) to obtain[3] the $t_{\mathrm{C}_{lin}}$:

$$t_{\mathrm{C}_{lin}} = \frac{\vec{r}_{\mathrm{A/B}} \cdot \vec{V}_{AB}}{\|\vec{V}_{AB}\|^2}. \tag{3.6.6}$$

**Linear Miss Distance** Figure 3.19b also shows the definition of the *minimum separation distance vector*, which is the negative of the vector rejection of $\vec{r}_{\mathrm{A/B}}$ from $\vec{V}_{AB}$,:

$$\vec{d}_{AB} = \frac{\vec{r}_{\mathrm{A/B}} \cdot \vec{V}_{AB}}{\|\vec{V}_{AB}\|^2}\vec{V}_{AB} - \vec{r}_{\mathrm{A/B}} \tag{3.6.7}$$

where $||\cdot||$ is the Euclidean norm operator and $\cdot$ is the dot product operator. The Euclidean norm of the minimum separation distance vector is defined as the linear miss distance:

$$d_{\mathrm{C}_{lin}} = \|\vec{d}_{AB}\|. \tag{3.6.8}$$

**Conflict Detection Conditions** It has been shown by Carbone *et al.* [2006] that the point $A$ and the sphere $B$ are headed for a collision if and only if the following conditions are satisfied:

$$\|\vec{d}_{AB}\| \leq R \quad \text{and} \quad \dot{r} < 0. \tag{3.6.9}$$

[3]This work used this formula, as the singularity only appears when $\|\vec{V}_{AB}\|^2$ is equal to zero.

In this work, the collision conditions are modified to incorporate the two alerting thresholds ($R_{TH}$ and $t_{TH}$) :

$$d_{\mathrm{C}_{lin}} \leq R_{TH} \quad \text{and} \quad 0 < t_{\mathrm{C}_{lin}} \leq t_{TH}. \tag{3.6.10}$$

Moreover a range test and an altitude test are used to introduce a trigger zone, only within which a conflict may be declared:

$$r \leq r_{TH} \quad \text{and} \quad |\hat{p}_{H,D} - \hat{p}_{I,D}| \leq \Delta h_{TH} \tag{3.6.11}$$

### 3.6.2 Conflict Resolution

Once the conflict detection conditions are satisfied, the *Minimum Deviation* control strategy, given by Luongo *et al.* [2009], is used to generate the conflict resolution command:

$$\boldsymbol{c}^d_{\mathrm{H}} = \begin{bmatrix} V^d_g & \chi^d & V^d_v \end{bmatrix}^T \tag{3.6.12}$$

where the superscript $^d$ denotes the desirable values derived from the $\vec{V}^{\,d}_A$, as shown in Figure 3.20b. The solutions for $\vec{V}^{\,d}_A$ in different situations will be described below.

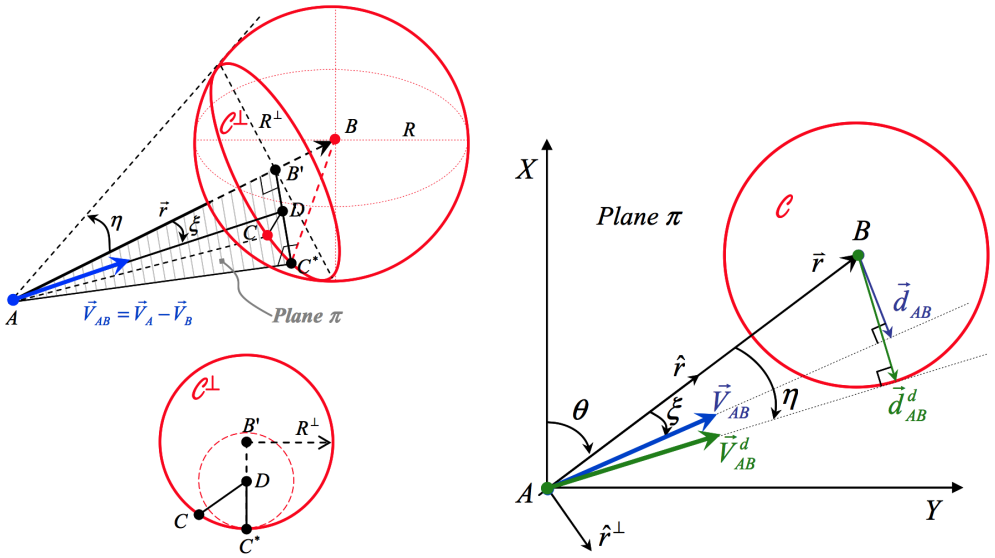
**Tangential Solution** With the assumption of no flight envelope limitations and dynamic constraints (along with the above linear assumption), Luongo *et al.* [2009] has shown that the below desired velocity is the optimal solution, in terms of deviations from the nominal (straight) route:

$$\vec{V}^{\,d}_A = \frac{\|\vec{V}_{AB}\| \cos(\eta - \xi)}{\sin \xi} \left[ \sin \eta \hat{V}_{AB} - \sin(\eta - \xi)\hat{\boldsymbol{r}} \right] + \vec{V}_B \tag{3.6.13}$$

with (see Figure 3.20a)

$$\xi = \arcsin \frac{\|\vec{\boldsymbol{d}}_{AB}\|}{\|\vec{\boldsymbol{r}}_{\mathrm{A/B}}\|} \quad \text{and} \quad \eta = \arcsin \frac{R'}{\|\vec{\boldsymbol{r}}_{\mathrm{A/B}}\|} \tag{3.6.14}$$

where the accent $\hat{}$ denotes the vector is a unit vector; and $R'$ is the radius of an *enlarged* safety bubble and serving as a tunable parameter for the algorithm, in order to compensate for the assumption of no dynamic constraints. The $R'$ is set to 2000 ft,




(a) Collision cone and the circumference of its base $\mathcal{C}^{\perp}$.

(b) The desired relative velocity vector on the collision plane $\boldsymbol{\pi}$.

Figure 3.20: Geometry of a conflict resolution solution, adapted from Luongo *et al.* [2009].

as suggested by Luongo *et al.* [2009], in order to protect the *target* safety bubble with radius of 500 ft. Note that, when $\xi$ is equal to zero, the solution in (3.6.13) becomes indeterminate and the following solution will be used.

**Constrained Solution** The above analytical solution (3.6.13), based on purely geometric consideration, does not take into account the aircraft's flight envelope limitations. If the desired velocity goes beyond the limitations (specified in B.1), i.e. $V_A^d \notin [V_{a,min}, V_{a,max}]$ or $V_{A,V}^d \notin [V_{V,min}, V_{V,max}]$[1], the *lateral-directional control strategy* given by Carbone *et al.* [2006] is used to generate the desired ground track angle $\chi_d$ while keeping the current ground speed $V_g^d$ and vertical speed $V_V^d$ constant. The generation of the desired ground track angle requires solving a fourth order polynomial, as explained by Carbone *et al.* [2006]; details are omitted here for brevity.

**Emergency Solution** When the separation distance between the point $A$ and sphere $B$ is less than $R'$ (i.e. the host is within the enlarged safety bubble), the solution in

[1] $V_{A,V}^d = -V_{A,D}^d$ is the vertical component of $\vec{V}_A^d$.

(3.6.13) cannot be adopted since there does not exist a tangent from the point $A$ to the sphere $B$ in this situation. The following emergency solution, as proposed by Luongo *et al.* [2009], will be adopted in this situation:

$$\vec{V}_A^{\,d} = -\|\vec{V}_A\|\hat{\boldsymbol{r}} \tag{3.6.15}$$

which provides a strong control action to exit from the enlarged safety bubble; after that the tangential solution (3.6.13) will continue to be applied.

### 3.6.3 Conflict Monitor

Once a conflict is detected, the collision avoidance logic initiates a process to monitor the evolution of the conflict so as to:

1. provide conflict resolution algorithms with an estimate of the intruder state when the intruder is temporarily beyond the host's FOR; this is achieved by maintaining a *conflict picture*;

2. trigger the corresponding transitions between the operating states of the collision avoidance logic, see Figure 3.18; this is achieved by continuously checking the transitions' guarding conditions, as summarized in Table 3.6

**Conflict Picture** Figure 3.21 shows the conflict picture of an example encounter. The conflict picture is initiated once a conflict is detected and will be maintained until the conflict is cleared, the period of which is referred to as a conflict window hereafter.

The figure also illustrates the three main elements of a conflict picture:

1. **Conflict coordinate system, denoted by** $\mathsf{C}$, is a *geographic coordinate system* with its origin at the host aircraft position where the conflict is first detected.

2. **Host trajectory** keeps track of the history of the host aircraft's positions, by converting the position information from the navigation system into the coordinates of the conflict coordinate system.

Table 3.6: Guarding conditions for the transitions among different finite states of the baseline collision avoidance logic.

| Transition | Guarding Conditions |
|---|---|
| Conflict Detected | • The linear time-to-CPA is greater than zero and smaller than a threshold;<br>• The linear miss distance is smaller than a threshold. |
| Conflict Resolved | • There exists no detected conflict;<br>• The range rate is greater than zero; and<br>• The range is greater than a threshold. |
| Clear of Conflict | •The conflict is resolved for a threshold period of time. |
| Back into FOR | •The output from the tracker system changes from invalid to valid. |
| Beyond FOR | • The output from the tracker system changes from valid to invalid. |

3. **Intruder trajectory** keeps track of the history of the intruder aircraft's positions, by converting the position information from the tracker system into the coordinates of the conflict coordinate system. Whenever there are no valid outputs from the tracker system, linear extrapolation based on the last position and velocity is used for estimation, see Figure 3.21.

## 3.7 Summary

In order to enable the capability to perform large-scale, fast-time Monte Carlo simulations under a wide range of testing conditions, this chapter has modelled the necessary components required to simulate the collision avoidance problems with non-cooperative sensing. The five models include 1) the Standard Encounter Model used to generate a credible set of testing scenarios; 2) a kinematic intruder model with Gaussian-noise manoeuvre commands; 3) an aircraft system model with configurable transient response characteristic and realistic aircraft performance; 4) a surveillance and tracking model able to produce the noisy and limited FOR conditions; and 5) a

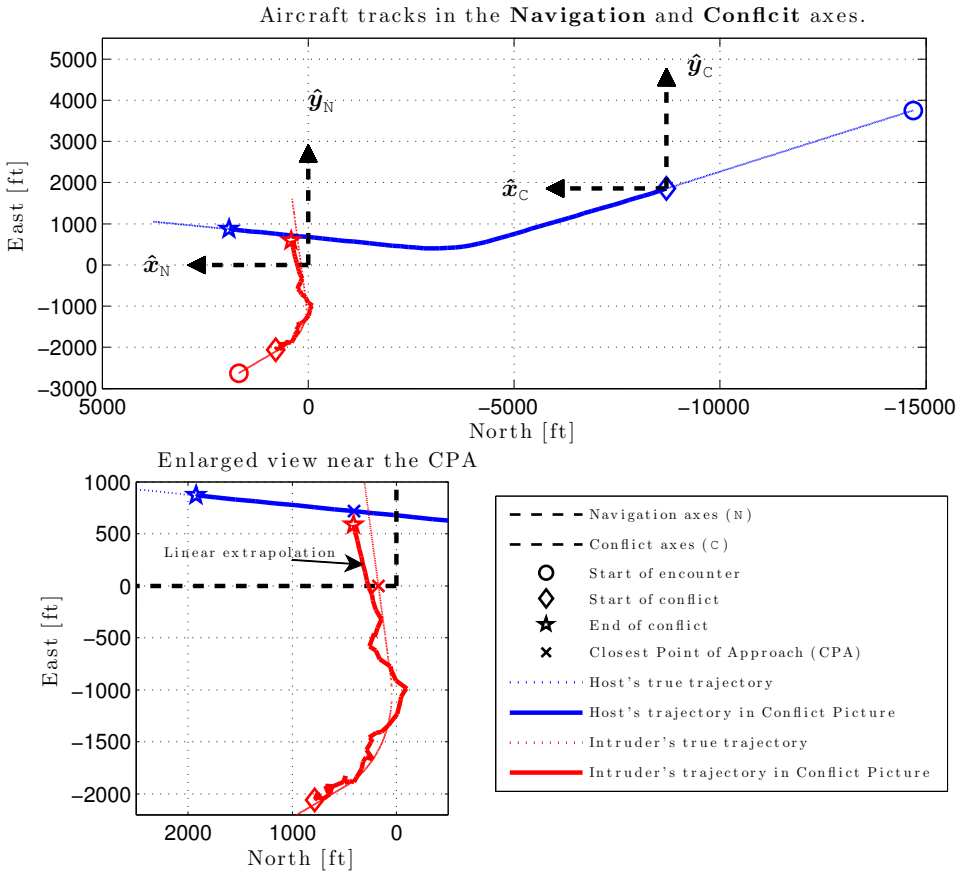



Figure 3.21: The conflict picture of an example encounter.

collision avoidance logic model serving as the baseline method.

The main result of this chapter is the modular and parametric modelling and simulation framework that can be used for large-scale, fast-time Monte Carlo simulations. The five functional modules, along with their interfaces and configurable parameters, are summarized in Tables 3.7 and 3.8.

This framework will be used to evaluate the performance of the baseline method in §4.4, and to perform a safety performance evaluation, robustness analysis and study of FOR effects in §6.3.

Table 3.7: Summary sheet of the models used for Monte-Carlo simulation (I).

| | Input | Output | Group | Configurable Parameter | Value | Unit |
|---|---|---|---|---|---|---|
| Encounter | No Inputs | Host Cmd. Intr. Cmd. | Encounter Set | Encounter ID. | $\{1,\ldots,100\}$ | - |
| | | | | Encounter class | $\{1,\ldots,19\}$ | - |
| | | | | Encounter type | $\{1,2\}$ | - |
| | | | Rejection Sampling | Host max. vertical speed | 1800 | ft/min |
| | | | | Host min. vertical speed | -2300 | ft/min |
| | | | | Host max. ground speed | 240 | kt |
| | | | | Host min. ground speed | 150 | kt |
| | | | | Intr. max. ground speed | 250 | kt |
| | | | | Intr. min. ground speed | 35 | kt |
| Intruder | Intr. Cmd. | Intr. States | Simulation | Manoeuvre uncertainty | $\{0,1\}$ | - |
| | | | Intruder Performance | Intr. max. vertical speed | 6000 | ft/min |
| | | | | Intr. min. vertical speed | -6000 | ft/min |
| | | | | Intr. max. ground speed | 400 | kt |
| | | | | Intr. min. ground speed | 0 | kt |
| | | | | Intr. max. turn rate | 50 | deg/s |
| | | | | Intr. min. turn rate | -50 | deg/s |
| Host | Host Manoeuvre Cmd. | Host States Navigation States | Aircraft | Performance model | Jetstream31 | - |
| | | | Envelope Protection | Max. airspeed | 290 | kt |
| | | | | Min. airspeed | 137 | kt |
| | | | | Max. vertical speed | 2000 | ft/min |
| | | | | Min. vertical speed | -2000 | ft/min |
| | | | Flight Control System | Max. airspeed rate | 2 | $ft/s^2$ |
| | | | | Min. airspeed rate | -2 | $ft/s^2$ |
| | | | | Max. turn rate | 9 | deg/s |
| | | | | Min. turn rate | -9 | deg/s |
| | | | | Max. flight-path angle rate | 0.582[a] | deg/s |
| | | | | Min. flight-path angle rate | -0.582 | deg/s |
| | | | Tracking Performance | Airspeed time constant | 5 | - |
| | | | | Heading time constant | 2 | - |
| | | | | Flight-path angle | 2 | - |

[a] Abbreviation: cmd. - command; intr. - intruder; max. - maximum; min. - minimum

[b] $0.582\,\text{deg/s} = \frac{5\,\text{ft/s}^2}{290\,\text{kt}} \times \frac{1}{1.688} \times \frac{180}{\pi}$, where $5\,\text{ft/s}^2$ is the maximum normal acceleration for civil flights, [see Nuic, 2012, p.33].

Table 3.8: Summary sheet of the models used for Monte-Carlo simulation (II).

| | Input | Output | Group | Configurable Parameter | Value | Unit |
|---|---|---|---|---|---|---|
| Surveillance | Intr. States | Tracker States | Simulation | Field of Regard switch | {0, 1} | - |
| | | | | Tracker uncertainty switch | {0, 1} | - |
| | | | | Tracker uncertainty level | {1, . . . , 5} | - |
| | | | FOR Limits | Relative range limit | 5 | NM |
| | | | | Relative azimuth limits | ±110 | deg |
| | | | | Relative elevation limits | ±15 | deg |
| | | | Tracker Performance | North position $\sigma$ | 50 | ft |
| | | | | East position $\sigma$ | 50 | ft |
| | | | | Down position $\sigma$ | 50 | ft |
| | | | | North speed $\sigma$ | 10 | ft/s |
| | | | | East speed $\sigma$ | 10 | ft/s |
| | | | | Down speed $\sigma$ | 10 | ft/s |
| Collision Avoidance Logic | Navigation States Tracker States | Resolution Cmd. | Conflict Detection | Linear time-to-CPA Th. | {15, 30} | s |
| | | | | Linear miss distance Th. | {500, . . . , 3000} | ft |
| | | | Conflict Resolution | Safety bubble radius | 2000 | ft |
| | | | | Max. inertial speed | 233 | kt |
| | | | | Min. inertial speed | 151 | kt |
| | | | | Max. vertical speed | 1968 | ft/min |
| | | | | Min. vertical speed | -1968 | ft/min |
| | | | Conflict Monitor | Min. range to clear a threat | 600 | ft |
| | | | | Min. time to confirm a clear of conflict | 2 | s |

[a] Abbreviation: cmd. - command; intr. - intruder; max. - maximum; min. - minimum; Th. - Threshold.

This page intentionally contains only this sentence.

# Chapter 4

# Evaluation and Analysis Framework

## 4.1 Introduction

With the modelling and simulations framework constructed in the previous chapter, the performance of a candidate collision avoidance logic can be evaluated via large-scale Monte Carlo simulations. However, given the enormous amount of simulation result data, a set of analysis tools (such as performance metrics and visualization tools) is required to facilitate the performance evaluation. The evaluation and analysis framework used throughout this thesis to evaluate the performance of collision avoidance logics is shown in Figure 4.1, in which the necessary analysis tools are described in the specified sections.

In the following, a typical set of metrics used to evaluate the collision avoidance logics' performance in terms of *safety* and *operational suitability* will be presented in §4.2. Furthermore, in order to enable an investigation of the FOR effect on the logic performance, an additional set of metrics is introduced in §4.3. Finally, the necessary analysis tool (such as System Operating Characteristic (SOC) and sensitivity curves) along with two simulation experiments—the establishment of the baseline performance and an investigation of the FOR effect on the logic performance—are described in §4.4.

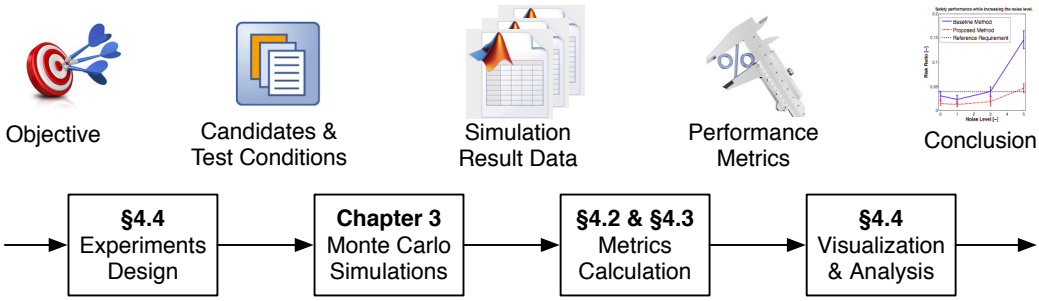



Figure 4.1: The evaluation and analysis framework used throughout this thesis.

## 4.2 Typical Performance Metrics

The aim of this section is to describe three typical performance metrics given by Kochenderfer *et al.* [2010a] [Raynaud & Arino, 2006; Holland *et al.*, 2013, see also]: *risk ratio*, *NMAC rate* and *unnecessary alert rate*[1]. §4.2.1 first describes a scheme used to categorize the possible outcomes of a simulated encounter; and then, based on the scheme, the mathematical definitions of the three metrics will be given in §4.2.2.

### 4.2.1 Possible Outcomes of an Encouter Scenario

There are six possible outcomes for a simulated encounter, depending on three basic events: (1) whether an NMAC occurred, (2) whether an alert was issued, and (3) whether an alert was necessary. The definitions of the three basic events are:

1. NMAC: an event defined to occur when separation between two aircraft is less than 100 ft apart vertically and 500 ft horizontally.

2. Alert: an alert issued by the collision avoidance logic when a conflict is detected.

3. Necessary Alert: an alert defined to be necessary when the original trajectory[2] results in an NMAC.

Figure 4.2 depicts all the possible outcomes using notional trajectories. The definitions and abbreviations for these events are summarized in Table 4.1.

[1]This is equivalent to the nuisance alert rate in this dissertation.
[2]The one without a collision avoidance system.

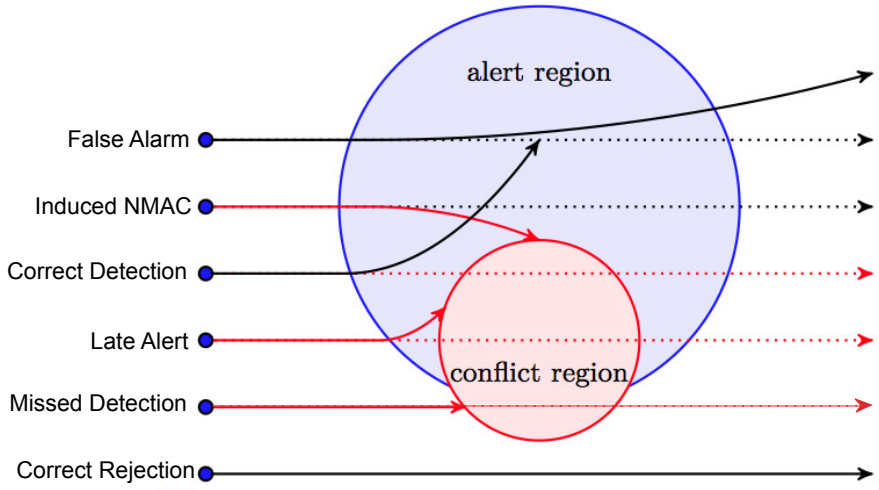



Figure 4.2: Notional illustration of all the possible outcomes of an encounter scenario. Solid lines represent the *avoidance trajectories* with the Collision Avoidance System (CAS), while dotted lines represent the *original trajectories* without the CAS, modified from Kochenderfer *et al.* [2010a, Figure 17].

Table 4.1: Event definitions for the possible outcomes of results.

| Outcomes Category | Abbreviation | Basic Events (Abbreviation) | | |
|---|---|---|---|---|
| | | Necessary Alert (NA) | System Alert (A) | NMAC (NM) |
| Correct Rejection | CR | | | |
| Correct Detection | CD | ✓ | ✓ | |
| False Alarm | FA | | ✓ | |
| Missed Detection | MD | ✓ | | ✓ |
| Induced NMAC | IN | | ✓ | ✓ |
| Late Alert | LA | ✓ | ✓ | ✓ |

* The italic version of an abbreviation represents the count of the respective event occurred in the simulation.
* The count of total encounter is $N = CR + CD + FA + MD + IN + LA$.
* The count of Necessary Alert is $NA = CD + MD + LA$.
* The count of Alert is $A = CD + FA + IN + LA$
* The count of NMAC is $NM = MD + IN + LA$

### 4.2.2 Metrics Definitions

Based on the counts of six outcomes for each simulation experiment, the following three performance metrics are defined[1]

1. **Risk ratio** is a measure of the change in the probability of NMAC due to the equipage of a collision avoidance logic. This is defined as:

$$RR \equiv \frac{\Pr(\text{NM}|\text{CDR})}{\Pr(\text{NM}|\overline{\text{CDR}})} = \frac{MD + IN + LA}{CD + MD + LA} = \frac{NM}{NA} \tag{4.2.1}$$

   where Pr(NM|CDR) is the probability of NMAC with the collision avoidance logic and Pr(NM|$\overline{\text{CDR}}$) is the one without the collision avoidance logic.

   A risk ratio of zero indicates that the logic resolves all NMAC, while a risk ratio of one indicates that the logic provides no benefit in reducing collision risk.

2. **NMAC rate**: The probability that an NMAC occurs can be estimated from the outcome counts as follows:

$$\Pr(\text{NM}) = \frac{MD + IN + LA}{CR + CD + FA + MD + IN + LA} = \frac{NM}{N} \tag{4.2.2}$$

   The smaller the NMAC rate, the better the system's safety performance in preventing collision is.

3. **Unnecessary alert rate**: The probability of unnecessary alert, Pr(UA), can be approximated by[1]:

$$\Pr(\text{UA}) = \frac{FA + IN}{CD + FA + IN + LA} = \frac{FA + IN}{A}. \tag{4.2.3}$$

   The smaller the unnecessary alert rate, the better the system's performance in detecting conflicts is.

A collision avoidance logic is deemed effective if risk ratio $RR$ and the probability of unnecessary alert rate Pr(UA) do not exceed the required level simultaneously.

[1]Note that, all metrics definitions in this section assume each encounter is equally likely while different types of encounters are normally not equally likely. For instance, ICAO [2007] gives a weighting for each encounter type in the Standard Encounter Model.

[1]Compared with Kochenderfer *et al.* [2010a], this definition excludes the $MD$ in the denominator.

## 4.3 Additional Metrics for Non-Cooperative Resolution

Two major issues for collision avoidance with non-cooperative intruders are the larger uncertainty in intruder state estimation and the possibility of losing sight of the intruder. In order to investigate the effect of these issues on the collision avoidance logics' performance, this section first discusses the basic concepts of these issues (§4.3.1) and then introduces four additional metrics for evaluation (§4.3.2).

### 4.3.1 Basic Concepts

Figure 4.3 illustrates the possible issues of non-cooperative collision avoidance: a) the field-of-regard status plot shows the possibility of losing sight of the intruder; b) the conflict resolution status plot depict the possibility of secondary conflict during an encounter; c) the intruder state plot shows the uncertainty in the state estimation. With the reference of Figure 4.3, the following features can be defined for each simulated encounter:

1. **Number of Detected Conflicts** $N_{con}$: the number of conflicts detected in a resolution encounter.

2. **Clear of Conflict** $\delta_{\mathrm{CoC}}$: a boolean variable indicating whether all the detected conflicts have been cleared before the simulation time ends.

3. **Feedback Availability** $\lambda_{ava}$: an index indicating the amount of feedback available when it is required; a feedback availability of one indicates that the feedback is always available, while a feedback availability of zero indicates that the feedback is not available at all.

   This can be estimated with the ratio of the number of time steps when the intruder is within the FOR to the number of time steps when the conflict is being resolved:

$$\lambda_{ava} = \sum_{j=1}^{N_{con}} \frac{\sum_{k=k_{cd,j}}^{k_{cc,j}} \delta_{FOR}(k)}{k_{cc,j} - k_{cd,j} + 1} \tag{4.3.1}$$

   where $\delta_{FOR}$ is a boolean variable indicating whether the intruder is within the FOR, $j$ is the index for detected conflicts, $k$ is the index for time steps, $k_{cd}$ and $k_{cc}$ are the time steps when the conflict is detected and cleared.

4. **Feedback Error** $\lambda_{err}$: an index indicating how accurate the estimations of the intruder states are, when they are required by the conflict resolution logics. This is calculated with the average of the Normalized Root Mean Square Error (NRMSE) of all the estimated intruder states:

$$\lambda_{ava} = \frac{\sum_{i=1}^{N_{sta}} NRMSE_i}{N_{sta}} \tag{4.3.2}$$

with

$$NRMSE_i = \sum_{j=1}^{N_{con}} \frac{RMSE_{i,j}}{x_{i,j,max} - x_{i,j,min}} \tag{4.3.3}$$

and

$$RMSE_{i,j} = \sqrt{\frac{\sum_{k=k_{cd,j}}^{k_{cc,j}} [\hat{x}_i(k) - x_i(k)]^2}{k_{cc,j} - k_{cd,j} + 1}} \tag{4.3.4}$$

where $x_i$ and $\hat{x}_i$ are the true and estimated values of the $i^{th}$ intruder state, $N_{sta}$ is the number of intruder states; $i$ is the index for intruder states.

### 4.3.2 Metrics Definitions

Based on the features defined in the previous subsection, four metrics are introduced:

1. **Secondary conflict rate**: An operational suitable collision avoidance logic should reduce the chance to cause a secondary conflict during the resolution. This can be measured by:

$$\Pr(\text{SC}) = \frac{SC}{A}, \tag{4.3.5}$$

where $SC$ is the count of simulated encounters in which the number of detected conflicts $N_{con}$ is larger than one.

2. **CoC (Clear of Conflict) rate**: An operational suitable collision avoidance logic should resolve the conflict in a reasonable time. The metric is defined as:

$$\Pr(\text{CoC}) = \frac{CoC}{A}, \tag{4.3.6}$$

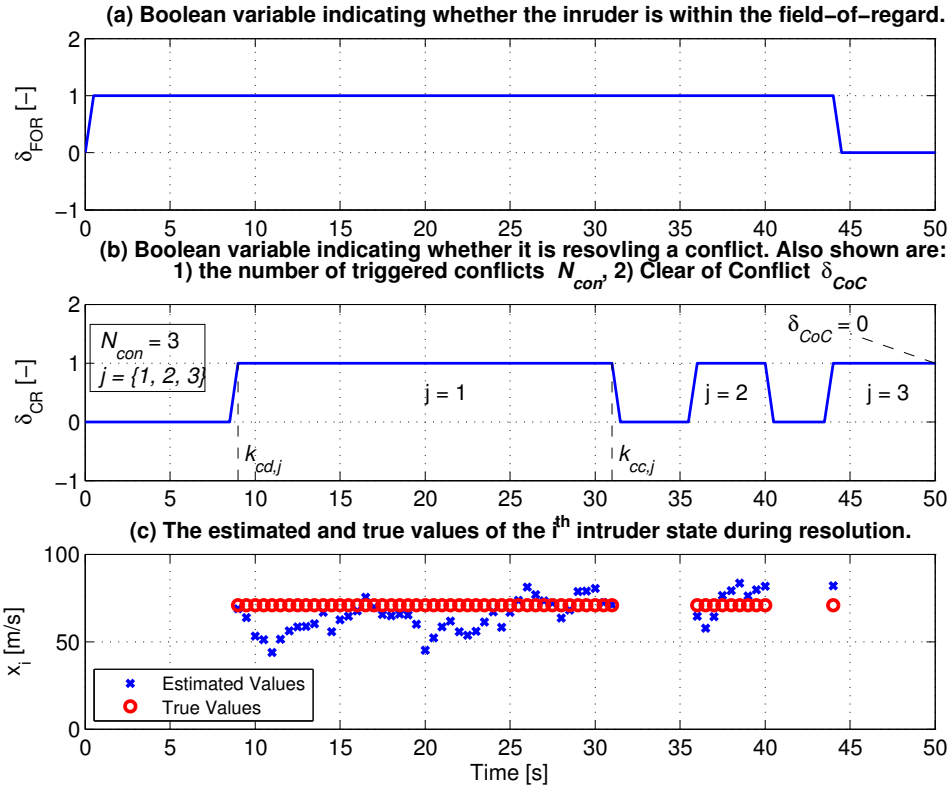



Figure 4.3: Illustrations of the basic concepts in conflict resolution.

where $CoC$ is the count of simulated encounters in which all the detected conflicts have been cleared.

3. **Expected feedback availability**: In order to evaluate the quality of feedback to the collision avoidance logic, the expected value of the feedback availability over all alerted encounters is used:

$$\mathrm{E}\left(\lambda_{ava}\right) = \frac{\sum_{\forall l=l_{\mathrm{A}}} \lambda_{ava,l}}{A}. \tag{4.3.7}$$

4. **Expected feedback error**: In order to evaluate the quality of feedback to the collision avoidance logic, the expected value of the feedback error index over all alerted encounters is used:

$$\mathrm{E}\left(\lambda_{err}\right) = \frac{\sum_{\forall l=l_{\mathrm{A}}} \lambda_{err,l}}{A}. \tag{4.3.8}$$

Table 4.2: Characteristics of the original aircraft trajectories in the standard encounter set.

| Characteristics | Intruder | | Host | |
|---|---|---|---|---|
| | Encounters | Rate (%) | Encounters | Rate (%) |
| Turning | 324 | 9 | 260 | 7 |
| Hori. Accelerating | 915 | 24 | 1664 | 44 |
| Veri. Accelerating | 3119 | 82 | 2065 | 54 |
| Manoeuvring* | 3304 | 87 | 2546 | 67 |

[1] $N = 3800$, $NM = 1615$, $\Pr(NM) = 0.425$.
[2] Encounters with both aircraft not manoeuvring = 227.
* Either turning, horizontally accelerating or vertically accelerating.

## 4.4 Two Simulation Experiments

Using the simulation experiment process given by Figure 4.1, this section presents two simulation experiments that were designed to:

1. establish a baseline performance for collision avoidance logics;
2. investigate the FOR effect on the performance of collision avoidance logics.

In the remainder of this section, the common experiment setting will be detailed in (§4.4.1); followed by an establishment of the baseline performance (§4.4.2) and an investigation of the FOR effect (§4.4.3).

### 4.4.1 Experiment Setting

**Standard Encounter Set** In order to achieve statistically meaningful results, a total of 3800 encounter samples were randomly generated using the Standard Encounter Model as described in §3.2.3. Table 4.2 summarizes the characteristic of the original aircraft trajectories in the standard encounter set. This standard encounter set will be used throughout the simulation experiments in this thesis.

**Test Conditions** There are twelve test conditions in total. They are characterized by different sensor models used in the simulations. Table 4.3 summarizes the simulation

Table 4.3: Test condition definitions.

| **Test Condition Name** | **Sensor Characteristics** | **FOR Switch** | **Tracker Error Switch** | **Tracker Error Level** |
|---|---|---|---|---|
| Cooperative-0 | Ideal | 0 | 0 | 0 |
| Cooperative -1 | Noisy Only | 0 | 1 | 1 |
| Cooperative-2 | Noisy Only | 0 | 1 | 2 |
| Cooperative-3 | Noisy Only | 0 | 1 | 3 |
| Cooperative-4 | Noisy Only | 0 | 1 | 4 |
| Cooperative-5 | Noisy Only | 0 | 1 | 5 |
| Non-Cooperative-0 | Limited-FOR Only | 1 | 0 | 0 |
| Non-Cooperative-1 | Limited-FOR and Noisy | 1 | 1 | 1 |
| Non-Cooperative-2 | Limited-FOR and Noisy | 1 | 1 | 2 |
| Non-Cooperative-3 | Limited-FOR and Noisy | 1 | 1 | 3 |
| Non-Cooperative-4 | Limited-FOR and Noisy | 1 | 1 | 4 |
| Non-Cooperative-5 | Limited-FOR and Noisy | 1 | 1 | 5 |

parameters setting and the terms used to describe the simulation results.

**Candidate Systems** In the following experiments, there is only one candidate collision avoidance logic, the geometric and reactive one as described in §3.6. However, in order to select a suitable set of alerting thresholds (see Table 3.8), twelve candidate configurations, as summarized in Table 4.4, will be tested.

The test conditions and candidates described in this section are obtained by setting the corresponding values of the model parameters given in Tables 3.7 and 3.8.

### 4.4.2 Establishment of the Baseline Performance

The establishment of the baseline performance takes three steps: 1) perform a trade-off study to select a suitable configuration; 2) evaluate the performance of the logic with the selected configuration; 3) evaluate the performance robustness to sensor noise of the logic with the selected configuration.

Table 4.4: Summary of twelve configurations for the candidate collision avoidance logic.

| Time Threshold = 15 [s] | | Time Threshold = 30 [s] | |
|---|---|---|---|
| **Candidate Name** | **Distance Threshold [ft]** | **Candidate Name** | **Distance] Threshold [ft]** |
| CAS-15-250 | 250 | CAS-30-250 | 250 |
| CAS-15-500 | 500 | CAS-30-500 | 500 |
| CAS-15-820 | 820 | CAS-30-820 | 820 |
| CAS-15-1640 | 1640 | CAS-30-1640 | 1640 |
| CAS-15-3280 | 3280 | CAS-30-3280 | 3280 |
| CAS-15-6560 | 6560 | CAS-30-6560 | 6560 |

**Trade-off Study** The purpose of the trade-off study is to select a suitable configuration for the collision avoidance logic so that the risk ratio requirement can be met with a reasonable unnecessary alert rate. The viability is determined by a set of performance requirements on safety, operational suitability and acceptability [Holland *et al.*, 2013]. For reference, ICAO [2007] sets for ACAS a logic risk ratio threshold of 0.04 and 0.18 when the intruder is and is not ACAS equipped, respectively. Although the performance requirements for the SAA are still under development, Cole *et al.* [2013] uses a risk ratio of 0.05 for the reference requirement. For the initial feasibility analysis in this thesis, a risk ratio of 0.05 is also chosen as a *notional requirement*.

The candidate logic was tested with 2 different time threshold values and 6 different distance threshold values, under 4 testing conditions. The results are summarized in 2 groups (time threshold) of 4 curves (testing conditions) with 6 points (distance threshold). These curves, as shown in Figure 4.4, are called the System Operating Characteristic (SOC) curves of the candidate logic [Kochenderfer *et al.*, 2010a]. The shape of the curve is traced out as the altering threshold is varying. Each point on the SOC curve, referred to as an *operating point*, corresponds to an alerting threshold setting specified in Table 4.4. Each operating point is estimated using 3800 simulated encounters and the resulting risk ratio is plotted against the unnecessary alert rate. The closer the operating points are to the lower-left conner, the better the system performs: i.e. a system achieves a low risk ratio while also maintain a low unnecessary alert rate.

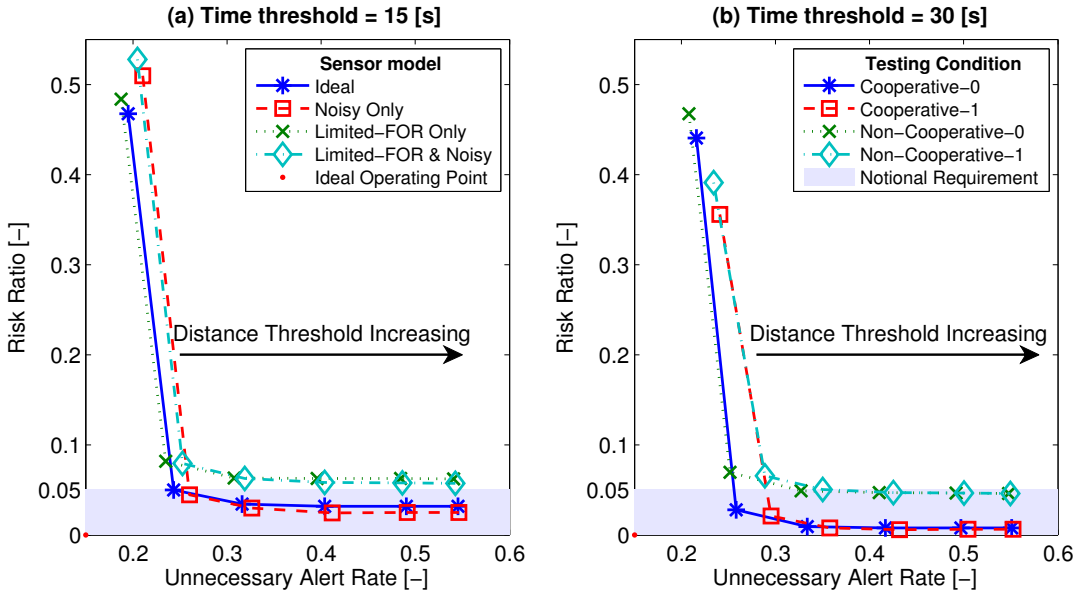



Figure 4.4: The System Operating Characteristic curves of the candidate collision avoidance logic under four testing conditions. This illustrates the trade-off between the risk ratio and unnecessary alert rate and shows the effect of the limited FOR and of the noise on the logic performance.

It can be observed from Figure 4.4 that:

- as the alerting thresholds are increasing, the operating points normally move to the lower-right corner of the graph, i.e. a safer operation can be achieved at the cost of more unnecessary alerts;
- compared to their ideal counterparts, the noisy operation points (denoted by the square and diamond markers) always achieved slightly better or comparable risk ratio while constantly having a larger unnecessary alert rate;
- the cooperative curves dominate the non-cooperative curves, i.e. with the same level of the unnecessary alert rate, the cooperative curves always achieve better lower risk ratio; and
- in order for the selected method to meet the notional requirement under all testing conditions (as shown in the four curves in the 30s time threshold plot), the unnecessary alert rate has to be traded for a lower risk ratio.

The baseline collision avoidance logic with the CAS-30-500 configuration is chosen as the *baseline configuration* for the remainder of the work in this thesis. This tight alerting threshold is deliberatively chosen so as to provide the performance evaluation of the proposed approach with more challenging situations; this will be shown in §6.3.

It is worth noting that, the sample size (3800) for this initial feasibility is very small, compared with those required by the ICAO [2007] guidance on ACAS or the study for the ACAS X presented by Cole *et al.* [2013].

**Performance Evaluation Statistics** Table 4.5 summarizes the evaluation results of the baseline configuration in the nominal noise level (noise level of 1). In addition to the three performance metrics, the table also reports the counts of three basic events and of six encounter outcomes to give an overview of the logic performance.

It can be seen that, the risk ratio is reduced from 1 to 0.0211 and to 0.0656[1] for the cooperative and non-cooperative case, respectively. This means that the logic is able to resolve about 97% and 93% of the original NMAC.

**Performance Robustness to Noise Level** The baseline configuration was tested in the 2 cases of cooperative and non-cooperative with 5 noise levels, leading to 10 sets of simulation results. Figures 4.5a and 4.5b summarise these results. These curves, referred to as sensitivity curves hereafter, illustrate the effect of varying the noise level on the probability of NMAC and on the unnecessary alert rate. Each point on the sensitivity curves was estimated using 4 independent sets of 950 simulated encounters (i.e. randomly divide 3800 simulated encounters into 4 sets). The error bars indicate the standard deviation of the estimates over the four sets.

As the noise level is increased from level 1 to 5, the NMAC and unnecessary alert rate (in the cooperative case) rise from 0.01 to 0.05 and from 0.3 to 0.375, respectively, indicating that the abilities of the system to prevent NMACs and to detect conflicts have degraded significantly. However, Figure 4.5a shows that the slope of the sensitivity curves appear relatively flatter when the noise level is small, and therefore the safety performance is relatively less sensitive to the small changes in noise level. Both the cooperative and non-cooperative curves are of similar shape, indicating that both cases'

[1]Note that, this does not satisfy the notional requirement shown in Figure 4.4, but the logic with the configuration of CAS-30-500 can provide a more challenging testing situation.

Table 4.5: Logic performance of the baseline configuration, under the nominal noise level.

| | Original | Cooperative | Non-Cooperative |
|---|---|---|---|
| **Risk Ratio** | 1.000 | 0.0211 | 0.0656 |
| **NMAC Rate** | 0.4250 | 0.0089 | 0.0279 |
| **Unnecessary Alert Rate** | - | 0.2954 | 0.2886 |
| **NMAC** | 1615 | 34 | 106 |
| **Necessary Alert** | - | 1615 | 1615 |
| **Issued Alert** | - | 2292 | 2197 |
| **Correct Rejection** | - | 1508 | 1551 |
| **Correct Detection** | - | 1585 | 1513 |
| **False Alarm** | - | 673 | 630 |
| **Missed Detection** | - | 0 | 52 |
| **Induced NMAC** | - | 4 | 4 |
| **Late Alert** | - | 30 | 50 |

[1] There are 3800 simulated encounters in total.

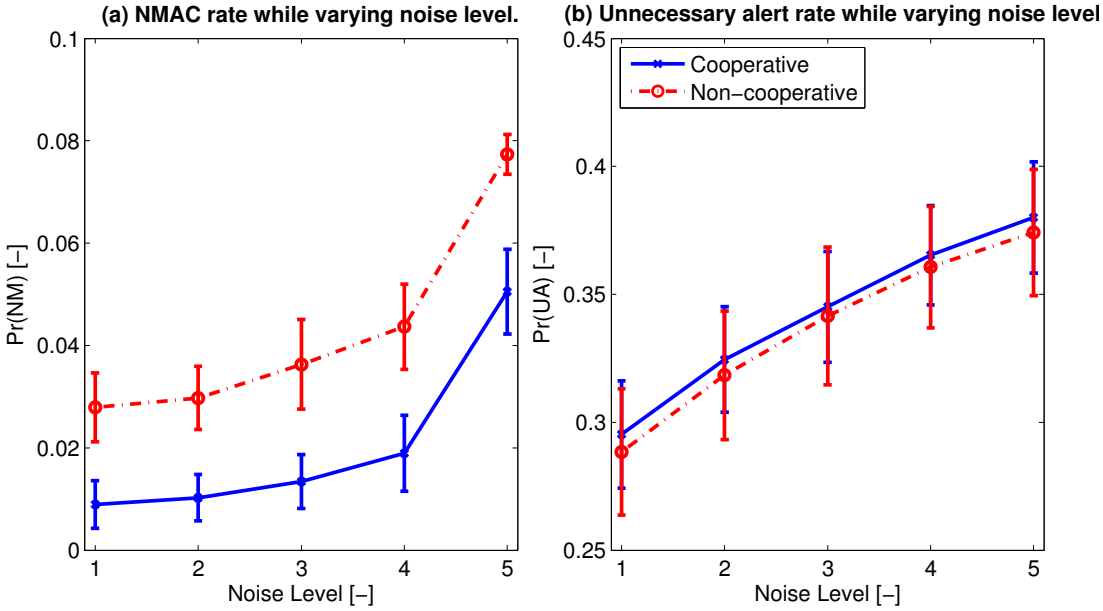



Figure 4.5: Evaluation of the baseline configuration's robustness to noise level.

sensitivity (and thus robustness) to the noise level are similar.

### 4.4.3 Investigation of Field-of-Regard Restriction Effect

In order to investigate the FOR on the logic performance, the baseline configuration was tested in the 2 cases (cooperative and non-cooperative) with 5 noise levels, leading to 10 sets of simulation results. For each set of the simulation result (consisting of 3800 encounters), the typical and additions metrics (as defined in §4.2 and §4.3) were evaluated. This subsection presents the evaluated metrics along with the analysis in three groups: safety performance, robustness and feedback quality.

**FOR Effect on Safety Performance** Figure 4.6 shows the bar chart of the resulting NMAC counts for each set of simulation results. Each bar comprise three groups of NMAC: Miss Detection, Induced NMAC and Unresolved NMAC. It can be seen that a salient FOR effect on safety performance is the increased numbers on NMACs due to missed detections. While the logic is able to detect almost all conflicts in the cooperative cases, there are about 50 missed detections in all five noise levels in the non-cooperative cases. Secondly, while the number of induced NMAC remains similar in two cases, there are more unresolved NMAC in the non-cooperative cases.

**FOR Effect on Feedback Quality** Figure 4.7a and 4.7b show the sensitivity curves of the expected feedback availability $\mathrm{E}(\lambda_{ava})$ and the expected feedback error $\mathrm{E}(\lambda_{err})$.

As can be seen in Figure 4.7a, the expected feedback availability drops from 1 in the cooperative cases to about 0.83 in the non-cooperative cases, indicating that, due to the limited-FOR, the host has lost sight of the intruder in about 17% of resolution duration. Moreover, the flat shape of the sensitivity curves shows that they are insensitive to the noise level.

Figure 4.7b shows that the expected feedback errors, in both cooperative and non-cooperative cases, are proportional to the noise level. In particular, when there is no noise at all, a significant difference in the feedback error can be found between the cooperative and non-cooperative situations. This difference, analogous to the effect of limited FOR, is a shrinking from 1 to about 0.3 as the noise level is growing.

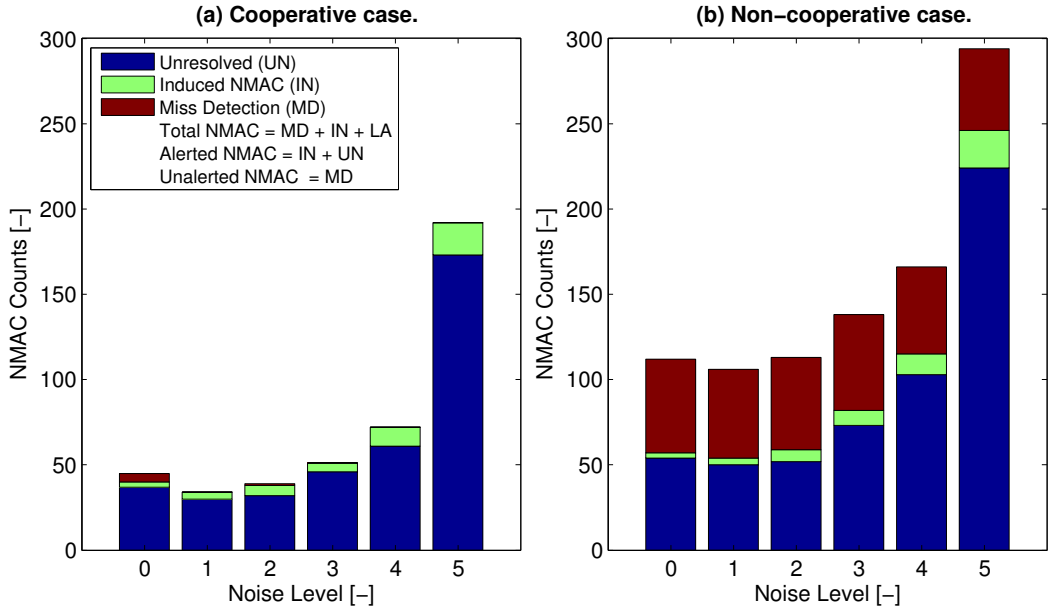



Figure 4.6: Composition of resulting NMAC events, illustrating the FOR effect on safety performance.

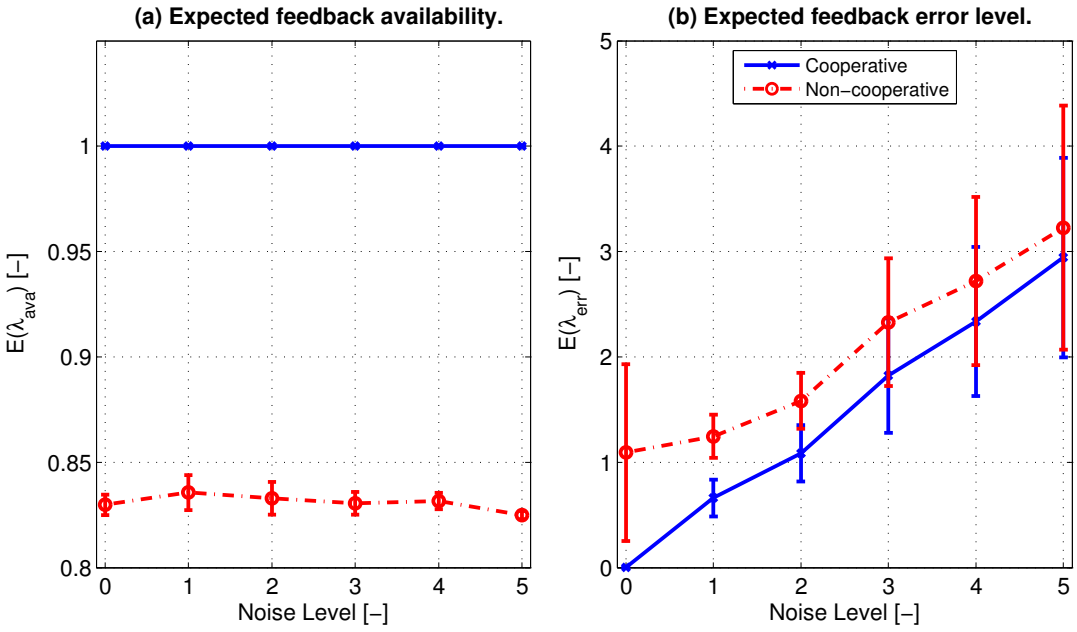



Figure 4.7: The resulting expected feedback quality metrics, illustrating the FOR effect on feedback quality.

**FOR Effect on Operational Suitability** Figures 4.8a and 4.8b show the sensitivity curves of the probability of clear of conflict Pr(CoC) and the probability of secondary conflict Pr(SC).

As can be seen from Figure 4.8a, the probability of clear of conflict Pr(CoC) is generally decreasing with the noise level; and the Pr(CoC) in non-cooperative cases are normally smaller than those in cooperative cases. These indicate that it is harder for the collision avoidance logic to clear a conflict in a noisier and/or non-cooperative situations.

Figure 4.8b shows that the probability of secondary conflict Pr(SC) is generally rising with the noise level; and, as the noise level increases, the rise in Pr(SC) in the cooperative cases become larger than those in non-cooperative cases, it is because there are significantly more chances for the collision avoidance logic to trigger the conflict conditions, by the noisy measurements which are available throughout the simulated encounter (compared with some absence of measurement in the non-cooperative cases).

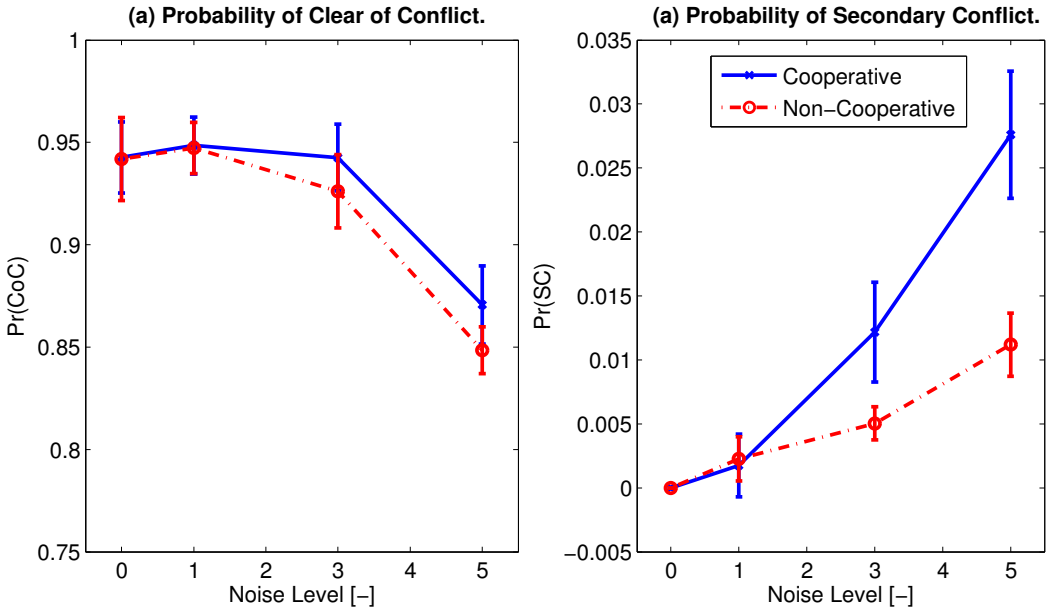



Figure 4.8: The resulting metrics for the probability of CoC and Secondary Conflict, illustrating the FOR effect on operational suitability.

## 4.5 Summary

This chapter has described a framework, consisting of a systematic process and a collection of analysis tools, for the performance evaluation and safety analysis of collision avoidance logic. In particular, a set of metrics is introduced to measure the operational suitability and feedback quality of the simulation results. Utilizing the framework, two simulation experiments have been carried out.

In the first experiment to establish a baseline performance, the following can be summarized:

- In the trade-off study, the reactive collision avoidance logic is shown to be effective at most operating points, i.e. the risk ratio and unnecessary alert rate normally do not exceed the required level simultaneously; however, in the non-cooperative cases, the reactive logic can only achieve the notional safety requirement by trading-off the unnecessary alert rate. This validates the vision stated in §1.1 that a more stringent unnecessary rate requirement for automated collision avoidance manoeuvres would invalidate some existing collision avoidance logics;

- A tight alerting threshold for the baseline collision avoidance logic (i.e. the reactive logic with the selected configuration) is deliberately chosen so as to provide the performance evaluation of the proposed approach (shown in §6.3) with more challenging situations. Despite the tight alerting threshold, the baseline logic is reasonably effective in preventing an NMAC, i.e. under the nominal noise level, it achieves a risk ratio of 0.0211 and 0.0656 (i.e. resolving about 97% and 93% of the original NMAC) in the cooperative and non-cooperative cases, respectively; and

- The baseline robustness performance has also been established.

In the second experiment to investigate the FOR effect on the logic performance, it has been quantitatively shown that:

- The immediate effect of the limited FOR is the reduction of the quality of feedback to the collision avoidance logic;

- The NMAC rate increases significantly as the result of limited FOR: when the noisy level is relatively small, most NMAC are caused by missed detections; however, as the noise level is increasing, the number of NMAC caused by miss detections remains steady but the number of Unsolved NMAC (caused by the more and more erroneous estimations of the intruder state) is rising rapidly with the noise level;
- The limited FOR makes it harder to determine the moment to issue a Clear of Conflict. This is manifested by the increase in the probability of CoC and the probability of Secondary Conflict.

In response to the above findings, the following will be further studied in the remainder of this thesis:

1. The baseline performance will be used for comparisons in Chapter 6 to investigate the potential benefit of the proposed approach;
2. The findings in the investigation of the FOR effect will be used in Chapter 5 to guide the requirements capture and problem formulation processes; and the FOR effect on logic performance will be revisited in §6.3.

# Chapter 5

# Trajectory Planning Algorithm Development

## 5.1 Introduction

The core of the proposed approach, as shown in §1.3, is the capability for a collision avoidance logic to plan a **flyable** and **operationally suitable** trajectory to **resolve an imminent collision threat** in **real time**. Specifically, in the context of this thesis, the following general requirements are defined:

- **Collision Avoidance**: Avoidance manoeuvres should achieve a minimum separation of 500 ft in the horizontal plane and 100 ft in the vertical plane from all other aircraft.[1]

- **Robust Collision Avoidance**: The performance in collision avoidance should be robust to the intruder state uncertainty.

- **Flyability**: All avoidance manoeuvre should be within the structural and aerodynamics performance limitations of the host aircraft at all flight conditions.[2]

- **Operational Suitability–Route Deviation**: The extent of the avoidance manoeuvres should be minimized, so that the host does not encroach on an adjacent

[1] See Sellem-Delmar [2010, HLR114]
[2] See F38 Committee [2007, §4.3.5.1].

altitude level or cause secondary conflicts with otherwise safely separated aircraft.[1]

- **Operational Suitability–Safe Terminal State**: Once the threat is resolved, the collision avoidance logic should (barring additional ATC or UAVp instructions) transmit clear of conflict to the UAVp and seek UAVp approval to conduct a "return-to-route" manoeuvre.[2] Therefore, the avoidance manoeuvre should terminate at a state that is *safe* and suitable for return-to-route.

This chapter describes the development of a Trajectory Planning Algorithm (TPA) for this capability. In the following, an avoidance trajectory planning problem accommodating the above requirements is first formulated in §5.2. The problem is then transcribed (converted) into an NLP problem via a direct method as described in §5.3. Finally, an NLP problem solver is presented in §5.4 to complete the algorithm. The testing results, some example trajectories and discussion will be shown in §5.5.

## 5.2 Problem Formulation

The above requirements for collision avoidance logics are incorporated into the following avoidance trajectory planning problem:

**Problem 5.2.1** (Trajectory Planning Problem (TPP))**.** *Find the system trajectory, consisting of the control and state trajectories:*

$$(\boldsymbol{u}^*(t), \boldsymbol{x}^*(t)) \,, \forall t \in \left[t_0, t_f\right] \tag{5.2.1}$$

*that minimizes the cost functional consisting of the final and trajectory cost:*

$$J = \phi_f \left(\boldsymbol{x}(t_f), \dot{\boldsymbol{x}}, \boldsymbol{u}\right) + \int_{t_0}^{t_f} L\left(\boldsymbol{u}(t), \boldsymbol{x}(t), t\right)\, dt \tag{5.2.2a}$$

*subject to the differential constraints:*

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right), \quad \forall t \in [t_0, t_f], \tag{5.2.2b}$$

[1] See Edwards [2012] and ICAO [2007, §4.4.4.3].

[2] See Sellem-Delmar [2010, HLR121 and HLR123], see also Lai & Whidborne [2012] for the discussion of the importance of a safe terminal state.

*the trajectory constraints:*

$$\boldsymbol{\psi}\left(\boldsymbol{x}(t),\dot{\boldsymbol{x}}(t),\boldsymbol{u}(t),\dot{\boldsymbol{u}},t\right) \leq 0, \quad \forall t \in [t_0,t_f], \tag{5.2.2c}$$

*the final constraints:*

$$\psi_f\left(\boldsymbol{x}(t_f),t_f\right) \leq 0, \tag{5.2.2d}$$

*and the initial conditions:*

$$\boldsymbol{x}\left(t_0\right) = \boldsymbol{x}_0. \tag{5.2.2e}$$

*The state and control evolutions are described by the functions (mappings):*

$$\boldsymbol{x}\left(\cdot\right): \left[t_0,t_f\right] \rightarrow \mathbb{R}^6, \quad \boldsymbol{u}\left(\cdot\right): \left[t_0,t_f\right] \rightarrow \mathbb{R}^3.$$

*The vector-value functions are the problem data and will be elaborated in §5.2.1-5.2.4:*

| | | |
|---|---|---|
| $\phi_f\left(\cdot\right)$ | $: \mathbb{R}^6 \times \mathbb{R}^6 \times \left[t_0,t_f\right] \rightarrow \mathbb{R}$ | *Safe Terminal State* |
| $L\left(\cdot\right)$ | $: \mathbb{R}^6 \times \mathbb{R} \rightarrow \left[t_0,t_f\right]$ | *Operational Suitability* |
| $f\left(\cdot\right)$ | $: \mathbb{R}^6 \times \mathbb{R}^3 \rightarrow \mathbb{R}^6$ | *Aircraft Dynamics* |
| $\boldsymbol{\psi}\left(\cdot\right)$ | $: \mathbb{R}^6 \times \mathbb{R}^6 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \left[t_0,t_f\right] \rightarrow \mathbb{R}^{17}$ | *Aircraft Performance and Collision Avoidance* |
| $\psi_f\left(\cdot\right)$ | $: \mathbb{R}^6 \times \left[t_0,t_f\right] \rightarrow \mathbb{R}$ | *Safe Terminal State* |

*where the initial time $t_0$ and final time $t_f$ are fixed.*

### 5.2.1 Differential Constraints

The differential constraints (system equations) are based on the aircraft dynamics (3.4.1). Considering the avoidance manoeuvre is subject to a 1) limited acceleration; 2) limited vertical rate; and 3) short duration (about 30 seconds), it is assumed that:

1. the wind disturbance is handled by an inner loop trajectory tracking controller (as will be seen in §6.2.3), so the wind-related terms in (3.4.1) will be dropped;

2. the flight-conditions-dependent parameters (such as $\rho(h)$, $C_{D0}(M)$) remain constant during the resolution manoeuvres, and the constant values are calculated according to the current flight condition (i.e. $\rho(h_0)$, $C_{D0}(M_0)$);

The position trajectory is described in a *resolution coordinate system*, which is a geographic coordinate system established at the current aircraft position according to the input to the TPA. Under the above assumptions, the aircraft's EoM (3.4.1) can be algebraically manipulated to give the following system equations:

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \dot{\boldsymbol{x}} = \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 \\ \dot{V} \\ \dot{\chi} \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} V\cos\gamma\cos\chi \\ V\cos\gamma\sin\chi \\ -V\sin\gamma \\ C_1\bar{T} - C_2V^2 + C_3\frac{n_z^2}{V^2} - g\sin\gamma \\ \frac{gn_z\sin\mu}{V\cos\gamma} \\ \frac{g(n_z\cos\mu-\cos\gamma)}{V} \end{bmatrix}, \tag{5.2.3}$$

with

$$\boldsymbol{x} = [\ p_1 \quad p_2 \quad p_3 \quad V \quad \chi \quad \gamma\ ]^T, \tag{5.2.4a}$$

$$\boldsymbol{u} = [\ \bar{T} \quad n_z \quad \mu\ ]^T, \tag{5.2.4b}$$

$$C_1 = \frac{C_{Tcr}\left(\frac{C_{Tc,1}}{V_0}\left(1-\frac{h_0}{C_{Tc,2}}\right)+C_{Tc,3}\right)}{m}, \tag{5.2.4c}$$

$$C_2 = \frac{S\rho(h_0)C_{D0}(M_0)}{2m}, \tag{5.2.4d}$$

$$C_3 = \frac{2mg^2k(M_0)}{\rho(h_0)S}, \tag{5.2.4e}$$

where $V$ and $\chi$ denote the ground speed and ground track, respectively; $V_0$ and $h_0$ denote the current airspeed and altitude and they are the input from the navigation system; and $p_1$, $p_2$, and $p_3$ denote the north, east and down coordinates in the resolution coordinate system; all other the variables are the same as those in (3.4.1),

### 5.2.2 Trajectory Constraints

**Aircraft Performance Limitations** Taking into account the limited aircraft performance (see Table 3.7), the following constraints are imposed during the resolution manoeuvres:

- Control variables:

$$\bar{T}_{min} \leq \bar{T} \leq \bar{T}_{max}, \quad \mu_{min} \leq \mu \leq \mu_{max}, \quad n_{z_{min}} \leq n_z \leq n_{z_{max}}(h_0, M_0). \quad (5.2.5)$$

- State variables:

$$V_{min}^{CR} \leq V \leq V_{max}^{CR}, \quad p_{3_{min}} \leq p_3 \leq p_{3_{max}}. \quad (5.2.6)$$

- Derivatives of control and state variables:

$$\dot{V}_{min} \leq \dot{V} \leq \dot{V}_{max}, \quad \dot{\mu}_{min} \leq \dot{\mu} \leq \dot{\mu}_{max}, \quad \dot{p}_{3_{min}}^{CR} \leq \dot{p}_3 \leq \dot{p}_{3_{max}}^{CR}. \quad (5.2.7)$$

The constraint values are obtained from the aircraft performance model, and can be found in Table B.1.

*Remark* 5.1. Note that, the constraint on the maximum angle of attack (3.4.6) is taken into account by the maximum normal load factor $n_{z_{max}}(h_0, M_0)$, whose value is dependent on the current flight condition (a lookup table for this dependency is implemented and shown in Figure B.2 in Appendix B).

**Collision Avoidance** To enforce the collision avoidance requirement, an imaginary collision volume is centred at the intruder aircraft. Figure 5.1 shows three possible models for the collision volume. The oblate spheroid model is used in this work because of its continuity over the cylinder model and its realism over the sphere model. The host aircraft is outside the intruder spheroid if and only if the following conditions holds:

$$\left(\frac{p_{\mathrm{I},1} - p_1}{a}\right)^2 + \left(\frac{p_{\mathrm{I},2} - p_2}{a}\right)^2 + \left(\frac{p_{\mathrm{I},3} - p_3}{b}\right)^2 > 1, \quad (5.2.8)$$

where $\boldsymbol{p}_{\mathrm{I}} = [p_{\mathrm{I},1} \quad p_{\mathrm{I},2} \quad p_{\mathrm{I},3}]^T$ denotes the intruder's position in the resolution coordinate system; $a$ and $b$ are the major and minor axis of the generating ellipse.

The selection of the axis lengths take into account the required separation and the estimated uncertainty (in intruder's position caused by the limited sensor accuracy, and in host's position caused by the imperfect avoidance manoeuvre execution). Figure 5.2 shows the construction of the generating ellipse of the oblate spheroid model. The

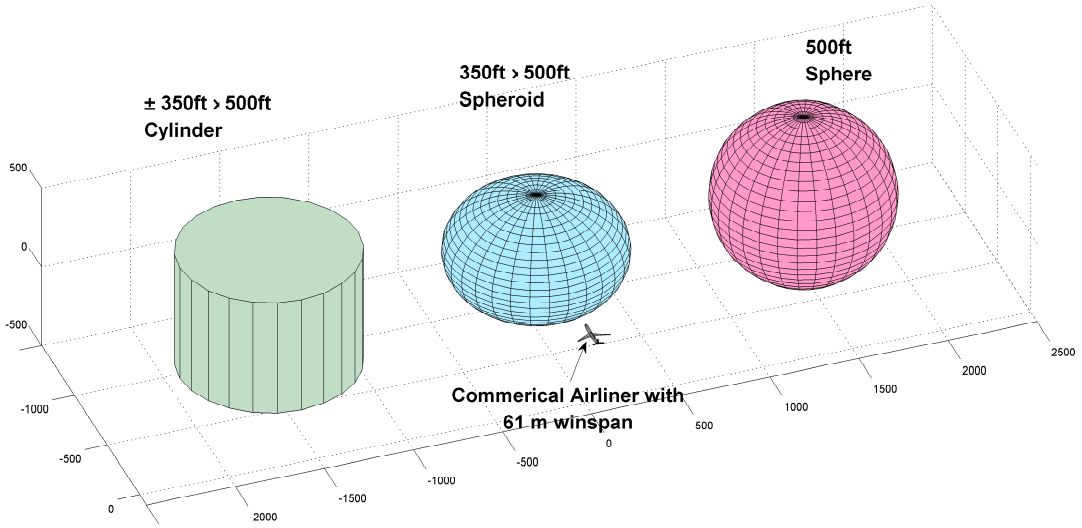



Figure 5.1: Three possible models for the protected volume.

lengths of the two axes are given by:

$$a_0 = \lambda\sqrt{2}R \quad \text{and} \quad b_0 = \lambda\sqrt{2}H, \tag{5.2.9}$$

where $R$ and $H$ are, respectively, the horizontal and vertical required separation; $\lambda$ is a design parameter according to the platform performance.

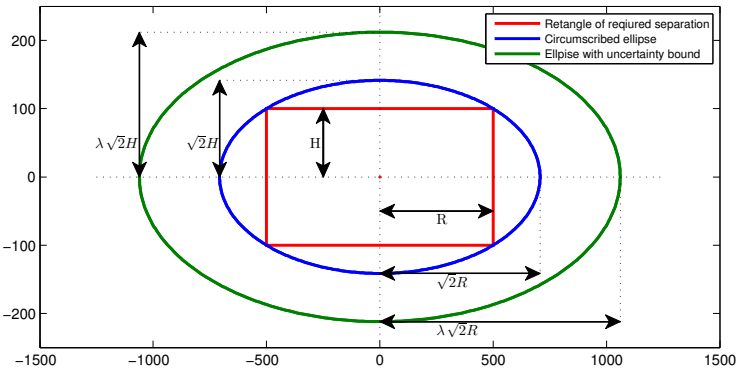



Figure 5.2: Construction of the generating ellipse of the oblate spheroid model of the collision volume.

The condition (5.2.8) is applied to a finite look-ahead time horizon, referred to as a *planning horizon*. This involves projecting the current intruder state into the future over the planning horizon. Kuchar & Yang [2000] identified three types of projection methods: nominal, probabilistic and worse-case.

This work uses an empirical approach to propagate the intruder uncertainty, as illustrated by Figure 5.3 .Firstly, using the linear extrapolation, it projects the intruder's estimated state to produce a *nominal* trajectory:

$$\boldsymbol{p}_{\mathtt{I}}(t) = \boldsymbol{p}_{\mathtt{I},0} + \dot{\boldsymbol{p}}_{\mathtt{I},0} t, \tag{5.2.10}$$

where the initial intruder position $\boldsymbol{p}_{\mathtt{I},0}$ and velocity $\dot{\boldsymbol{p}}_{\mathtt{I},0}$ are the input to the trajectory generator. Secondly, in order to compensate for the prediction uncertainty, it enlarges the safety margin exponentially with time along with the nominal trajectory:

$$a(t) = a_0 \cdot g^{t/\tau} \quad \text{and} \quad b(t) = b_0 \cdot g^{t/\tau} \tag{5.2.11}$$

where $a_0$ and $b_0$ denote, respectively, the major and minor axis of the spheroid at the initial time; the growth factor $g$ and time constant $\tau$ are the design parameters.

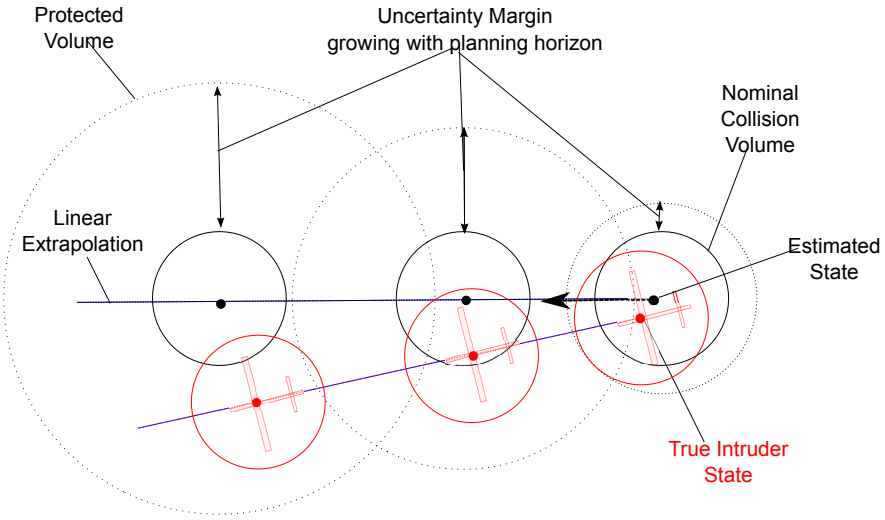



Figure 5.3: An empirical approach to propagate the intruder uncertainty.

**Trajectory Constraints** In summary, the trajectory constraints in Problem 5.2.1 consist of the performance (control, state, derivative) and obstacle constraint functions:

$$\boldsymbol{\psi}\left(\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{u},\dot{\boldsymbol{u}},t\right)=\begin{bmatrix}\boldsymbol{\psi}_c\left(\boldsymbol{u}\right)\\ \boldsymbol{\psi}_s\left(\boldsymbol{x}\right)\\ \boldsymbol{\psi}_d\left(\dot{\boldsymbol{x}},\dot{\boldsymbol{u}}\right)\\ \psi_o\left(\boldsymbol{x},t\right)\end{bmatrix}, \tag{5.2.12}$$

with

$$\boldsymbol{\psi}_c\left(\boldsymbol{u}\right)=\begin{bmatrix}\bar{T}-\bar{T}_{max}\\ \bar{T}_{min}-\bar{T}\\ \mu-\mu_{max}\\ \mu_{min}-\mu\\ n_z-n_{z_{max}}\left(h_0,M_0\right)\\ n_{z_{min}}-n_z\end{bmatrix},\quad \boldsymbol{\psi}_s\left(\boldsymbol{x}\right)=\begin{bmatrix}p_3-p_{3_{max}}\\ p_{3_{min}}-p_3\\ V-V_{max}\\ V_{min}-V\end{bmatrix},\quad \boldsymbol{\psi}_d\left(\dot{\boldsymbol{x}},\dot{\boldsymbol{u}}\right)=\begin{bmatrix}\dot{p}_3-\dot{p}_{3_{max}^{CR}}\\ \dot{p}_{3_{min}}^{CR}-\dot{p}_3\\ \dot{V}-\dot{V}_{max}^{CR}\\ \dot{V}_{min}^{CR}-\dot{V}\\ \dot{\mu}-\dot{\mu}_{max}\\ \dot{\mu}_{min}-\dot{\mu}\end{bmatrix},$$

$$\psi_o(\boldsymbol{x},t)=1-\left(\frac{p_1-p_{\mathrm{I},1}(t)}{a(t)}\right)^2-\left(\frac{p_2-p_{\mathrm{I},2}(t)}{a(t)}\right)^2-\left(\frac{p_3-p_{\mathrm{I},3}(t)}{b(t)}\right)^2. \tag{5.2.13}$$

### 5.2.3 Boundary Conditions

**Initial Conditions** At the initial time ($t_0 = 0$) when the trajectory generator is engaged, the initial position is initialized as zero, and the initial ground seed, ground track and flight path angle are initialized using the current values from the navigation system:

$$\boldsymbol{x}_0=\begin{bmatrix}p_{1,0} & p_{2,0} & p_{3,0} & V_0 & \chi_0 & \gamma_0\end{bmatrix}^T=\begin{bmatrix}0 & 0 & 0 & V_{nav} & \chi_{nav} & \gamma_{nav}\end{bmatrix}^T. \tag{5.2.14}$$

**Final Constraints** The final time ($t_f$) for the resolution manoeuvre is fixed at the planning horizon and served as a design parameter in this work.[1]

To ensure the resolution manoeuvre would terminate at a state that is safe, the final state of the resolution manoeuvre $\boldsymbol{x}\left(t_f\right)$ is constrained so that the relative range rate $\dot{r}$ is increasing, see (3.6.4):

$$\dot{r} = \frac{\boldsymbol{p}_r\left(t_f\right) \cdot \boldsymbol{v}_r\left(t_f\right)}{||\boldsymbol{p}_r\left(t_f\right)||} > 0, \tag{5.2.15}$$

where the relative position and velocity vector are:

$$\boldsymbol{p}_r(t) = \boldsymbol{p}_{\mathrm{I}}(t) - \boldsymbol{p}(t), \quad \boldsymbol{v}_r(t) = \dot{\boldsymbol{p}}_{\mathrm{I}} - \dot{\boldsymbol{p}}(t). \tag{5.2.16}$$

with the intruder trajectory $\boldsymbol{p}_{\mathrm{I}}(t)$ (5.2.10) as a known function.

Therefore, the final constraint function in Problem 5.2.1 is:

$$\psi_f\left(\boldsymbol{x}, t_f\right) = -\frac{\sum_{i=1}^{3}\left(p_{\mathrm{I},i}(t_f) - p_i\right)\left(\dot{p}_{\mathrm{I},i}(t_f) - \dot{p}_i\right)}{\sqrt{\sum_{i=1}^{3}\left(p_{\mathrm{I},i}(t_f) - p_i\right)^2}}. \tag{5.2.17}$$

**Final Cost** It is desirable that the resolution manoeuvre would terminate at a state that is suitable for the return-to-route manoeuvres; and for this purpose, we consider two factors: being parallel to the original route and being non-accelerating:

1. The final velocity is parallel to the original route, when:

$$\chi(t_f) = \chi_{rou}, \quad \gamma(t_f) = \gamma_{rou}, \tag{5.2.18}$$

   where $\chi_{rou}$ and $\chi_{rou}$ are the ground track and flight path angle of the original route at the planning horizon.

[1]Note that, this condition is not necessary for the proposed method and the final time can be set free and determined by the optimizer.

2. It is in a wing-level, straight and non-accelerating flight, when:

$$\mu(t_f) = 0, \quad \dot{\chi}(t_f) = 0, \quad \dot{\gamma}(t_f) = 0, \quad \dot{V}(t_f) = 0. \tag{5.2.19}$$

Therefore, the final cost function in Problem 5.2.1 is:

$$\phi_f(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}) = \left|\frac{\chi(t_f) - \chi_{rou}}{\chi_{rou}}\right| + \left|\frac{\gamma(t_f) - \gamma_{rou}}{\gamma_{rou}}\right| + \left|\mu(t_f)\right| + \left|\dot{\chi}(t_f)\right| + \left|\dot{\gamma}(t_f)\right| + \left|\dot{V}(t_f)\right|. \tag{5.2.20}$$

### 5.2.4 Cost Functions

The operational suitability of the generated trajectory is modelled by the cost functions. For instances, the required *route deviations* from the original route is a typical operational cost function. In this work, considering the non-cooperative nature of the problem, two tentative cost functions, *converging time* and *intruder invisibility*, are introduced.

**Route Deviation** In order to reduce the chances of causing secondary conflict, the deviation from the original route should be minimized. The square of the distance between the aircraft and the route position can be calculated with:

$$L_1(\boldsymbol{x}(t), t) = ||\left(\boldsymbol{p}(t) - \boldsymbol{p}_{rou}(t)\right)||_2^2 \tag{5.2.21}$$

where the original route coordinate $\boldsymbol{p}_{rou}(t)$ is a known function, as the input to the trajectory planning algorithm. Its definite integral can be used as a cost function to penalize the route deviation:

$$\int_0^{t_f} L_1(\boldsymbol{x}(t), t)\, dt. \tag{5.2.22}$$

**Negative Range Rate (Converging) Duration/Magnitude** Two aircraft are heading to a collision course if and only if their relative range rate is negative. Therefore, minimizing the duration/magnitude of negative range rate should reduce the risk of

collision.[1] As illustrated in Figure 5.4a, the negative part of the range rate function can be obtained by the following piecewise function:

$$L_2\left(\boldsymbol{x}(t),t\right) = \begin{cases} -\dot{r}(t) & \text{if } \left(\boldsymbol{p}_r(t)\cdot\boldsymbol{v}_r(t)\right) < 0, \\ 0 & \text{otherwise.} \end{cases}, \tag{5.2.23}$$

where $\boldsymbol{p}_r$ and $\boldsymbol{v}_r$ are the relative position and velocity (5.2.16); and its definite integral can be used to penalize the converging duration/magnitude:

$$\int_0^{t_f} L_2\left(\boldsymbol{x}(t),t\right)\, dt. \tag{5.2.24}$$

**Intruder Invisibility** When the intruder is beyond the FOR, the feedback to the conflict resolution logic is cut off. Therefore, it may be safer to choose a resolution manoeuvre that can safely keep the intruder within the FOR, rather than other options that do not take this into account. Recall that, in order to estimate whether the intruder is within the FOR, see §3.5.2, the host aircraft's attitude is required:

$$\phi = \mu, \qquad \theta = \alpha + \gamma, \qquad \psi = \chi, \tag{5.2.25}$$

where $\mu$, $\gamma$ and $\chi$ can be obtained from the system equation (5.2.3) and the angle of attack $\alpha$ is estimated, according to (3.4.4) and (3.4.5), with:

$$\alpha = \frac{1}{C_{L_\alpha}}\left(\frac{2n_z mg}{\rho V^2 S} - C_{L_0}\right). \tag{5.2.26}$$

With the host's attitude and equations (3.5.9)-(3.5.11), the intruder's relative position vector, expressed in the spherical coordinate system $\mathsf{B_s}$, can be calculated:

$$\boldsymbol{p}_{\mathtt{I}/\mathsf{B_s}} = \begin{bmatrix} r_{\mathsf{B_s}} & \Phi_{\mathsf{B_s}} & \Theta_{\mathsf{B_s}} \end{bmatrix}^T, \tag{5.2.27}$$

[1]Note that, special care should be given to this cost function. This should only apply to feasible (collision-free) trajectories; as, in some cases, an infeasible trajectory that moves toward to the intruder as soon as possible may get a minimum converging duration trajectory.

and then a *Field Of Regard* function can be defined:

$$\delta_{FOR}(t) = \begin{cases} 1 & \text{if } \left(r_{\mathsf{B_s}} \leq r_{max}\right) \wedge \left(\left|\Phi_{\mathsf{B_s}}\right| \leq \Phi_{lim}\right) \wedge \left(\left|\Theta_{\mathsf{B_s}}\right| \leq \Theta_{lim}\right) \\ 0 & \text{otherwise.} \end{cases} \tag{5.2.28}$$

Furthermore, as illustrated in Figure 5.4b, let $\delta_{NRR}$ be a *Negative Range Rate function*:

$$\delta_{NRR}(t) = \begin{cases} 1 & \text{if } \left(\boldsymbol{p}_r(t) \cdot \boldsymbol{v}_r(t)\right) < 0 \\ 0 & \text{otherwise,} \end{cases} \tag{5.2.29}$$

and $L_3$ be the following piecewise function:

$$L_3\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right) = \begin{cases} 1 & \text{if } \delta_{NRR}(t) \wedge \delta_{FOR}(t), \\ 0 & \text{otherwise,} \end{cases} \tag{5.2.30}$$

then the definite integral of $L_3$:

$$\int_0^{t_f} L_3\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right)\, dt \tag{5.2.31}$$

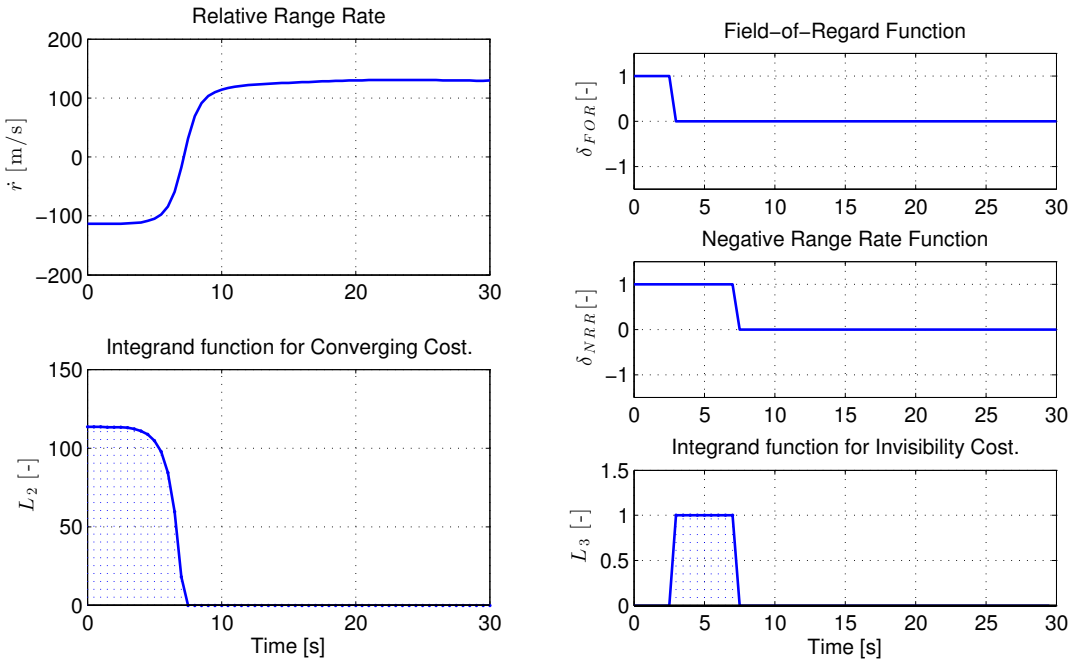
can be used to penalize any avoidance manoeuvres that takes the intruder outside the FOR; and this penalty only applies when two aircraft are converging.

## 5.3 Problem Transcription

This section describes a direct method, based on that in Yakimenko [2000], to transcribe the Problem 5.2.1 into a 15-dimension NLP (Problem 5.3.1). The transcription process involves four steps: removing the differential constraints, parameterizing the trajectory, satisfying the boundary conditions, and discretizing the constraint and cost functions.

### 5.3.1 Differential Constraint Removal

The differential flatness property of dynamical systems is used to remove the differential constraint in Problem 5.2.1.




(a) Converging cost function.

(b) Invisibility cost function.

Figure 5.4: Illustration of the Converging and Invisibility Cost Functions..

**Differential Flatness** According to Murray [2010, Definition 1.1], a nonlinear system:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{5.3.1}$$

is differentially flat if there exists a function $\boldsymbol{z}(\cdot)$:

$$\boldsymbol{z} = \boldsymbol{z}\left(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, \ldots, \boldsymbol{u}^{(p)}\right) \tag{5.3.2}$$

such that we can write the solution of the nonlinear system (5.3.1) as functions of the *flat output* $\boldsymbol{z}$ and a finite number of its derivatives $\boldsymbol{z}^{(q)}$:

$$\boldsymbol{x} = \boldsymbol{x}\left(\boldsymbol{z}, \dot{\boldsymbol{z}}, \ldots, \boldsymbol{z}^{(q)}\right), \tag{5.3.3}$$

$$\boldsymbol{u} = \boldsymbol{u}\left(\boldsymbol{z}, \dot{\boldsymbol{z}}, \ldots, \boldsymbol{z}^{(q)}\right). \tag{5.3.4}$$

*Remark* 5.2. This definition is sufficient for our application and a more rigorous treatment to the flatness property can be found in Fliess *et al.* [1995].

The significance of a system being differentially flat is that all system behaviour ($\boldsymbol{x}$ and $\boldsymbol{u}$) can be expressed without integration by the flat outputs $\boldsymbol{z}$ and a finite number of its derivatives. In other words, the system equation (and thus the differential constraints) (5.3.1) will be automatically satisfied when varying the flat output.

**Inverse Dynamics Model** Recall that, the system equation (5.2.3) for Problem 5.2.1 is in the form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}), \quad \text{with} \quad \boldsymbol{x} = \begin{bmatrix} p_1 & p_2 & p_3 & V & \chi & \gamma \end{bmatrix}^T, \quad \boldsymbol{u} = \begin{bmatrix} \bar{T} & n_z & \mu \end{bmatrix}^T. \tag{5.3.5}$$

Choosing the aircraft position as the flat output, i.e. $\boldsymbol{z} \triangleq \boldsymbol{p}$, the system equation (5.2.3) can be rearranged to give the flat output as a function of the state:

$$\boldsymbol{p} = \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix}^T \triangleq \boldsymbol{p}(\boldsymbol{x}), \tag{5.3.6}$$

the state as a function of the flat output and its first derivative $\boldsymbol{x}(\boldsymbol{p},\dot{\boldsymbol{p}})$:

$$\boldsymbol{x} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ V \\ \chi \\ \gamma \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \sqrt{\dot{p_1}^2 + \dot{p_2}^2 + \dot{p_3}^2} \\ \operatorname{atan2}(\dot{p_2}, \dot{p_1}) \\ -\arctan \frac{\dot{p_3}}{\sqrt{\dot{p_1}^2 + \dot{p_2}^2}} \end{bmatrix} \triangleq \boldsymbol{x}(\boldsymbol{p},\dot{\boldsymbol{p}}), \tag{5.3.7}$$

and the control as a function of the first and second derivation of the flat output $\boldsymbol{u}(\dot{\boldsymbol{p}},\ddot{\boldsymbol{p}})$:

$$\boldsymbol{u} = \begin{bmatrix} \bar{T} \\ n_z \\ \mu \end{bmatrix} = \begin{bmatrix} \arctan \frac{V\dot{\chi}\cos\gamma}{V\dot{\gamma}+g\cos\gamma} \\ \frac{1}{g}\sqrt{(V\dot{\gamma} + g\cos\gamma)^2 + (V\dot{\chi}\cos\gamma)^2} \\ \frac{1}{C_1}\left(\dot{V} + C_2 V^2 + g\,\sin\gamma - \frac{C_3 {n_z}^2}{V^2}\right) \end{bmatrix} \triangleq \boldsymbol{u}(\dot{\boldsymbol{p}},\ddot{\boldsymbol{p}}) \tag{5.3.8}$$

with

$$
\begin{aligned}
\cos\gamma &= \sqrt{\frac{\dot{p_1}^2 + \dot{p_2}^2}{\dot{p_1}^2 + \dot{p_2}^2 + \dot{p_3}^2}},\\
\sin\gamma &= \frac{\dot{p_3}}{\sqrt{\dot{p_1}^2 + \dot{p_2}^2}},\\
\dot{V} &= \frac{\dot{p}_1\ddot{p}_1 + \dot{p}_2\ddot{p}_2 + \dot{p}_3\ddot{p}_3}{\sqrt{\dot{p_1}^2 + \dot{p_2}^2 + \dot{p_3}^2}},\\
\dot{\chi} &= \frac{\dot{p}_1\ddot{p}_2 - \ddot{p}_1\dot{p}_2}{\dot{p}_1^2 + \dot{p}_2^2},\\
\dot{\gamma} &= \frac{\dot{p}_3\left(\dot{p}_1\ddot{p}_1 + \dot{p}_2\ddot{p}_2\right) - \ddot{p}_3\left(\dot{p}_1^2 + \dot{p}_2^2\right)}{\left(\dot{p}_1^2 + \dot{p}_2^2 + \dot{p}_3^2\right)\sqrt{\dot{p}_1^2 + \dot{p}_2^2}}.
\end{aligned}
$$

Similarly, the functions $\dot{\boldsymbol{x}}\,(\dot{\boldsymbol{p}},\ddot{\boldsymbol{p}})$ and $\dot{\boldsymbol{u}}\,(\ddot{\boldsymbol{p}},\dddot{\boldsymbol{p}})$ can also be obtained by taking the derivatives of (5.3.7) and (5.3.8). The functions $\boldsymbol{x}\,(\boldsymbol{p},\dot{\boldsymbol{p}})$, $\dot{\boldsymbol{x}}\,(\dot{\boldsymbol{p}},\ddot{\boldsymbol{p}})$, $\boldsymbol{u}\,(\dot{\boldsymbol{p}},\ddot{\boldsymbol{p}})$ and $\dot{\boldsymbol{u}}\,(\ddot{\boldsymbol{p}},\dddot{\boldsymbol{p}})$, are referred to as the *inverse dynamics model* hereafter, as they are inverting the system dynamics to give the control and state from the output.

Therefore, with the functions inverse dynamics model, the differential constraints in Problem 5.2.1 can be removed and the problem now is to find the optimal position trajectory and its derivatives, i.e. $\boldsymbol{p}, \dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}}$.

### 5.3.2 Parameterization

Each position coordinate is parameterized with a global seven-degree polynomial:

$$p_j(t) = a_{j0} + a_{j1}t + \frac{a_{j2}}{2}t^2 + \frac{a_{j3}}{6}t^3 + \frac{a_{j4}}{12}t^4 + \frac{a_{j5}}{20}t^5 + \frac{a_{j6}}{30}t^6 + \frac{a_{j7}}{42}t^7, \tag{5.3.9a}$$

$$\dot{p}_j(t) = a_{j1} + a_{j2}t + \frac{a_{j3}}{2}t^2 + \frac{a_{j4}}{3}t^3 + \frac{a_{j5}}{4}t^4 + \frac{a_{j6}}{5}t^5 + \frac{a_{j7}}{6}t^6, \tag{5.3.9b}$$

$$\ddot{p}_j(t) = a_{j2} + a_{j3}t + a_{j4}t^2 + a_{j5}t^3 + a_{j6}t^4 + a_{j7}t^5, \tag{5.3.9c}$$

$$\dddot{p}_j(t) = a_{j3} + 2a_{j4}t + 3a_{j5}t^2 + 4a_{j6}t^3 + 5a_{j7}t^4 \tag{5.3.9d}$$

$$\forall j = 1,2,3$$

in which, $a_{jk}$ is the $k^{th}$ coefficient of the $j^{th}$ coordinate.

The coefficients are defined by the boundary values of the coordinates and their

derivatives. They can be determined by the following set of linear equations:

$$\boldsymbol{a}_j = \underline{\mathbf{C}}^{-1}\boldsymbol{b}_j, \quad j = 1,2,3 \tag{5.3.10}$$

in which $\underline{\mathbf{C}}$ is the constant matrix depending on the initial and final time ($t = t_0 = 0$ and $t = t_f$), $\boldsymbol{a}_j$ is the coefficient vector for each coordinate and $\boldsymbol{b}_j$ is a vector containing all the boundary values:

$$\underline{\mathbf{C}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & t_f & \frac{t_f^2}{2} & \frac{t_f^3}{6} & \frac{t_f^4}{12} & \frac{t_f^5}{20} & \frac{t_f^6}{30} & \frac{t_f^7}{42} \\ 0 & 1 & t_f & \frac{t_f^2}{2} & \frac{t_f^3}{3} & \frac{t_f^4}{4} & \frac{t_f^5}{5} & \frac{t_f^6}{6} \\ 0 & 0 & 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 0 & 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \end{bmatrix}, \quad \boldsymbol{a}_j = \begin{bmatrix} a_{j0} \\ a_{j1} \\ a_{j2} \\ a_{j3} \\ a_{j4} \\ a_{j5} \\ a_{j6} \\ a_{j7} \end{bmatrix}, \quad \boldsymbol{b}_j = \begin{bmatrix} p_{j0} \\ \dot{p}_{j0} \\ \ddot{p}_{j0} \\ \dddot{p}_{j0} \\ p_{jf} \\ \dot{p}_{jf} \\ \ddot{p}_{jf} \\ \dddot{p}_{jf} \end{bmatrix}$$

By solving the linear equation set (5.3.10), the coefficients can be expressed in terms of the boundary values $\boldsymbol{b}_j = \begin{bmatrix} p_{j_0} & \dot{p}_{j_0} & \ddot{p}_{j_0} & \dddot{p}_{j_0} & p_{j_f} & \dot{p}_{j_f} & \ddot{p}_{j_f} & \dddot{p}_{j_f} \end{bmatrix}$ and the final time $t_f$:

$$\begin{aligned}
a_{j0} &= p_{j0}, \quad a_{j1} = \dot{p}_{j0}, \quad a_{j2} = \ddot{p}_{j0}, \quad a_{j3} = \dddot{p}_{j0}, \\
a_{j4} &= -\frac{2\dddot{p}_{jf} + 8\dddot{p}_{j0}}{t_f} + \frac{30\ddot{p}_{jf} - 60\ddot{p}_{j0}}{t_f^2} - \frac{180\dot{p}_{jf} + 240\dot{p}_{j0}}{t_f^3} + 420\frac{p_{jf} - p_{j0}}{t_f^4}, \\
a_{j5} &= \frac{10\dddot{p}_{jf} + 20\dddot{p}_{j0}}{t_f} - \frac{140\ddot{p}_{jf} - 200\ddot{p}_{j0}}{t_f^2} + \frac{780\dot{p}_{jf} + 900\dot{p}_{j0}}{t_f^3} - 1680\frac{p_{jf} - p_{j0}}{t_f^4}, \\
a_{j6} &= -\frac{15\dddot{p}_{jf} + 20\dddot{p}_{j0}}{t_f} + \frac{195\ddot{p}_{jf} - 225\ddot{p}_{j0}}{t_f^2} - \frac{1020\dot{p}_{jf} + 1080\dot{p}_{j0}}{t_f^3} + 2100\frac{p_{jf} - p_{j0}}{t_f^4}, \\
a_{j7} &= 7\frac{\dddot{p}_{jf} + \dddot{p}_{j0}}{t_f} - 84\frac{\ddot{p}_{jf} - \ddot{p}_{j0}}{t_f^2} + 420\frac{\dot{p}_{jf} + \dot{p}_{j0}}{t_f^3} - 840\frac{p_{jf} - p_{j0}}{t_f^4}.
\end{aligned} \tag{5.3.11}$$

Therefore, the parameterized position trajectory is completely determined by its

boundary values, namely the initial and final positions, velocities, accelerations and jerks.

### 5.3.3 Boundary Conditions Satisfaction

By pre-satisfying the given boundary conditions, the dimension of the problem can be further reduced. Recall that, the velocity can be calculated with its spherical coordinates (5.2.3):

$$\dot{p}_1 = V\cos\gamma\cos\chi, \quad \dot{p}_2 = V\cos\gamma\sin\chi, \quad \dot{p}_3 = -V\sin\gamma, \tag{5.3.12}$$

and the acceleration can be obtained by taking the derivatives of them:

$$\ddot{p}_1 = \dot{V}\cos\chi\cos\gamma - \dot{\chi}V\cos\gamma\sin\chi - \dot{\gamma}V\cos\chi\sin\gamma \tag{5.3.13a}$$

$$\ddot{p}_2 = \dot{V}\cos\gamma\sin\chi + \dot{\chi}V\cos\chi\cos\gamma - \dot{\gamma}V\sin\chi\sin\gamma \tag{5.3.13b}$$

$$\ddot{p}_3 = -\dot{V}\sin\gamma - \dot{\gamma}V\cos\gamma \tag{5.3.13c}$$

**Initial Conditions** To use the initial conditions (5.2.14) with the velocity formula (5.3.12), the initial position and velocity can be obtained:

$$\boldsymbol{p}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \quad \text{and} \quad \dot{\boldsymbol{p}}_0 = \begin{bmatrix} V_0\cos\gamma_0\cos\chi_0 & V_0\cos\gamma_0\sin\chi_0 & -V_0\sin\gamma_0 \end{bmatrix}^T \tag{5.3.14a}$$

To enable a smoother transit from the current state to the resolution trajectory, the initial acceleration (of the resolution trajectory) is set to the values obtained from the navigation system:

$$\ddot{\boldsymbol{p}}_0 = \begin{bmatrix} a_N & a_E & a_D \end{bmatrix}^T. \tag{5.3.14b}$$

Note that, this initial acceleration condition will be modified later in the integration section §6.2.1.

The initial jerk is completely free and determined by the optimizer:

$$\dddot{\boldsymbol{p}}_0 = \begin{bmatrix} \dddot{p}_{10} & \dddot{p}_{20} & \dddot{p}_{30} \end{bmatrix}^T. \tag{5.3.15}$$

**Final Conditions** The final position, velocity, acceleration and jerk are not fixed and determined by the optimizer:

$$\boldsymbol{p}_f = \begin{bmatrix} p_{1f} & p_{2f} & p_{3f} \end{bmatrix}^T, \quad \ddot{\boldsymbol{p}}_f = \begin{bmatrix} \ddot{p}_{1f} & \ddot{p}_{2f} & \ddot{p}_{3f} \end{bmatrix}^T, \quad \dddot{\boldsymbol{p}}_f = \begin{bmatrix} \dddot{p}_{1f} & \dddot{p}_{2f} & \dddot{p}_{3f} \end{bmatrix}^T. \tag{5.3.16}$$

In particular, using the velocity's polar coordinates (5.3.12), the final velocity is parameterized as:

$$\dot{\boldsymbol{p}}_f = \begin{bmatrix} V_f \cos\gamma_f \cos\chi_f \\ V_f \cos\gamma_f \sin\chi_f \\ -V_f \sin\gamma_f \end{bmatrix}, \tag{5.3.17}$$

where $V_f$, $\gamma_f$, and $\chi_f$ are the final ground speed, flight path angle, and ground track and will be determined by the optimizer.

In summary, with the equations (5.3.9)-(5.3.17), the aircraft position, velocity, acceleration and jerk functions can be parameterized as followed:

$$\boldsymbol{p}(t) \triangleq \boldsymbol{p}(\boldsymbol{\Xi}, t), \quad \dot{\boldsymbol{p}}(t) \triangleq \dot{\boldsymbol{p}}(\boldsymbol{\Xi}, t), \quad \ddot{\boldsymbol{p}}(t) \triangleq \ddot{\boldsymbol{p}}(\boldsymbol{\Xi}, t), \quad \dddot{\boldsymbol{p}}(t) \triangleq \dddot{\boldsymbol{p}}(\boldsymbol{\Xi}, t), \forall t \in [t_0, t_f]. \tag{5.3.18}$$

with an *optimization vector* with fifteen optimization variables:

$$\boldsymbol{\Xi} = \begin{bmatrix} V_f & p_{1f} & p_{2f} & p_{3f} & \dddot{p}_{1f} & \dddot{p}_{2f} & \dddot{p}_{3f} & \dddot{p}_{10} & \dddot{p}_{20} & \dddot{p}_{30} & \ddot{p}_{1f} & \ddot{p}_{2f} & \ddot{p}_{3f} & \gamma_f & \chi_f \end{bmatrix}^T. \tag{5.3.19}$$

*Remark* 5.3. The order of the optimization variables may have some effects on the optimization process for some optimization algorithms (such as the Hooke-Jeeves pattern search we used in §5.4.4), we used this particular order throughout this thesis and our implementation.

### 5.3.4 Trajectory Discretization

**Parameterized Constraint and Cost Integrand Functions** After the removal of differential constraints and parameterization of the position trajectory, the constraint functions and the integrand functions of cost functions can be expressed as functions

of the optimization vector and time, for instance:

$$\boldsymbol{\psi}\left(\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{u},\dot{\boldsymbol{u}},t\right) \xrightarrow[\boldsymbol{p}(\boldsymbol{\Xi},t),\dot{\boldsymbol{p}}(\boldsymbol{\Xi},t),\ddot{\boldsymbol{p}}(\boldsymbol{\Xi},t),\dddot{\boldsymbol{p}}(\boldsymbol{\Xi},t)}{\boldsymbol{x}(\boldsymbol{p},\dot{\boldsymbol{p}}),\dot{\boldsymbol{x}}(\dot{\boldsymbol{p}},\ddot{\boldsymbol{p}}),\boldsymbol{u}(\dot{\boldsymbol{p}},\ddot{\boldsymbol{p}}),\dot{\boldsymbol{u}}(\ddot{\boldsymbol{p}},\dddot{\boldsymbol{p}})} \boldsymbol{\psi}\left(\boldsymbol{\Xi},t\right), \quad \forall t \in [t_0,t_f], \tag{5.3.20}$$

$$L_1\left(\boldsymbol{x}(t),t\right) \xrightarrow[\boldsymbol{p}(\boldsymbol{\Xi},t),\dot{\boldsymbol{p}}(\boldsymbol{\Xi},t),\ddot{\boldsymbol{p}}(\boldsymbol{\Xi},t),\dddot{\boldsymbol{p}}(\boldsymbol{\Xi},t)}{\boldsymbol{x}(\boldsymbol{p},\dot{\boldsymbol{p}}),\dot{\boldsymbol{x}}(\dot{\boldsymbol{p}},\ddot{\boldsymbol{p}}),\boldsymbol{u}(\dot{\boldsymbol{p}},\ddot{\boldsymbol{p}}),\dot{\boldsymbol{u}}(\ddot{\boldsymbol{p}},\dddot{\boldsymbol{p}})} L_1\left(\boldsymbol{\Xi},t\right). \tag{5.3.21}$$

**Collocation Points and Constraint Vector** To translate the optimal control problem into a finite-dimensional NLP problem, it is necessary to discretize the time interval $[t_0,t_f]$ into $N-1$ intervals with $N$ *collocation points*. The $N$ collocation points, where the constraints will be satisfied, are chosen uniformly over the time interval $[t_0,t_f]$:

$$\boldsymbol{t} = \left[t_1,t_2,\ldots,t_N\right]^T, \quad t_i = t_0 + (i-1)\frac{t_f}{N-1}, \quad i = 1,\ldots,N. \tag{5.3.22}$$

Let $\boldsymbol{c}$ be the constraint vector for the resulting NLP problem. The constraint vector consist of the trajectory and final constraint evaluated at the collocation points

$$\boldsymbol{c}\left(\Xi\right) = \begin{bmatrix} \boldsymbol{\psi}\left(\Xi,t_1\right) \\ \vdots \\ \boldsymbol{\psi}\left(\Xi,t_N\right) \\ \boldsymbol{\psi}_f\left(\Xi,t_N\right) \end{bmatrix} \in \mathbb{R}^{17N+1} \tag{5.3.23}$$

**Quadrature Rules for Cost Function** Recall that, in numerical analysis, the definite integral of a function $f(t)$ can be approximated using the *extended trapezoidal rule*:

$$\int_a^b f\left(t\right)\, dt \approx \frac{h}{2}\sum_{k=1}^{N-1}\left(f\left(t_{k+1}\right) + f\left(t_k\right)\right), \quad h = \frac{b-a}{N-1}. \tag{5.3.24}$$

Let $F$ be the cost function for a NLP problem; and apply the above extended trapezoidal rule to calculate the integrals of the three cost functions in §5.2.4:

$$F_i\left(\boldsymbol{\Xi}\right) = \frac{t_f - t_0}{2(N-1)}\sum_{j=1}^{2}\sum_{k=1}^{N-1}\left(L_j\left(\boldsymbol{\Xi},t_{k+1}\right) + L\left(\boldsymbol{\Xi},t_k\right)\right), \quad i = 1,2,3. \tag{5.3.25}$$

Furthermore, the final costs (5.2.20) are evaluated at the final time to give fourth components of the cost function $F$:

$$F_4(\Xi) = \phi_f(\boldsymbol{\Xi}, t_N). \tag{5.3.26}$$

Therefore, the cost function of Problem 5.2.1 is transcribed into the following cost function for the NLP problem:

$$F(\boldsymbol{\Xi}) = \sum_{i=1}^{4} F_i(\boldsymbol{\Xi}) \tag{5.3.27}$$

To sum up, the transcribed NLP problem can now be written as:

**Problem 5.3.1** (Nonlinear Programming Problem)**.** *Find*

$$\boldsymbol{\Xi}^* = \arg\min_{\boldsymbol{\Xi}\in\mathbb{S}} F(\boldsymbol{\Xi}), \tag{5.3.28}$$

*subject to:*

$$\boldsymbol{c}(\boldsymbol{\Xi}) \leq \boldsymbol{0}, \tag{5.3.29}$$

*where $\mathbb{S}$ denotes the solution space and will be discussed in §5.4.1.*

## 5.4 Problem Solution

This section presents four core elements for our NLP solver: an analysis of the solution space of the NLP problem; the scaling of the NLP problem for better performance and constraint satisfaction accuracy; the penalty function used to handle the constraints in the NLP; and finally the NLP algorithm used for optimization.

### 5.4.1 Solution Space

Recall that, the optimization variable vector is:

$$\boldsymbol{\Xi} = [V_f \quad p_{1f} \quad p_{2f} \quad p_{3f} \quad \dddot{p}_{1f} \quad \dddot{p}_{2f} \quad \dddot{p}_{3f} \quad \dddot{p}_{10} \quad \dddot{p}_{20} \quad \dddot{p}_{30} \quad \ddot{p}_{1f} \quad \ddot{p}_{2f} \quad \ddot{p}_{3f} \quad \gamma_f \quad \chi_f]^T,$$

and therefore, the solution space is the 15-dimensional vector space over the field of the real numbers, i.e. $\mathbb{S} = \mathbb{R}^{15}$. Prior to performing a solution search, the solution space can be reduced to a smaller search space by considering the feasibility of the given problem. In our cases, all the optimization variables are the boundary values of an aircraft trajectory and therefore can be bounded according to their physical properties.

**Feasibility on the Boundaries** The reduction of the search space can be achieved by studying the constraint vector ($\boldsymbol{\psi}\,(\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{u},\dot{\boldsymbol{u}},t)$) and the inverse dynamics model ($\boldsymbol{x}\,(\boldsymbol{p},\dot{\boldsymbol{p}})$, $\dot{\boldsymbol{x}}\,(\dot{\boldsymbol{p}},\ddot{\boldsymbol{p}})$, $\boldsymbol{u}\,(\dot{\boldsymbol{p}},\ddot{\boldsymbol{p}})$ and $\dot{\boldsymbol{u}}\,(\ddot{\boldsymbol{p}},\dddot{\boldsymbol{p}})$). For instance, the final vertical (down coordinate) rate constraint:

$$\dot{p}_{3max} \geq \dot{p}_{3,f} \geq \dot{p}_{3min} \xrightarrow{\dot{p}_{3,f}=-V_f \sin\gamma_{rou}} \begin{cases} -\frac{\dot{p}_{3min}}{\sin\gamma_{rou}} \geq V_f \geq -\frac{\dot{p}_{3max}}{\sin\gamma_{rou}} & \text{if } \gamma_{rou} \geq 0, \\ -\frac{\dot{p}_{3max}}{\sin\gamma_{rou}} \geq V_f \geq -\frac{\dot{p}_{3min}}{\sin\gamma_{rou}} & \text{if } \gamma_{rou} < 0, \end{cases} \tag{5.4.1}$$

can be combined to the final speed constraint:

$$V_{f\,max} \geq V_f \geq V_{f\,min} \xrightarrow[\dot{p}_{3max}>0]{\dot{p}_{3min}<0} V_{f\,max}' \geq V_f \geq V_{f\,min}, \tag{5.4.2}$$

with

$$V_{f\,max}' = \begin{cases} \min\left(V_{f\,max}, -\frac{\dot{p}_{3min}}{\sin\gamma_{rou}}\right) & \text{if } \gamma_{rou} \geq 0, \\ \min\left(V_{f\,max}, -\frac{\dot{p}_{3max}}{\sin\gamma_{rou}}\right) & \text{if } \gamma_{rou} < 0. \end{cases}$$

Therefore, a trajectory with its final speed satisfying (5.4.2) will also have a feasible vertical rate.

On the other hand, given the initial position $\boldsymbol{p}_0$, velocity $\dot{\boldsymbol{p}}_0$, and acceleration $\ddot{\boldsymbol{p}}_0$ (from the navigation system), the initial elements (except those related to the initial bank rate) of the constraint $\boldsymbol{c}\,(\boldsymbol{\Xi},t)$ (5.3.23) can also be calculated using the inverse dynamics model. When there exists any element that is greater than zero, Problem (5.3.1) is initially infeasible. In the simulation experiments, this initial infeasibility did happen, the handling and discussion of which will be deferred until §5.5.3.

**Reduced Solution Space** The Optimization Variables (OVs) are subject to the following lower and upper bound:

- 1st OV: the final speed is subject to the speed limits (5.4.2):

$$V_f{}'_{max} \geq V_f \geq V_{f\,min} \tag{5.4.3}$$

- 2nd – 4th OVs: the final position are expressed with its cylindrical coordinate to facilitate the imposition of the reachable limits. Figure 5.5 illustrates the final position's coordinate transformation and limit imposition. Therefore, the opti-

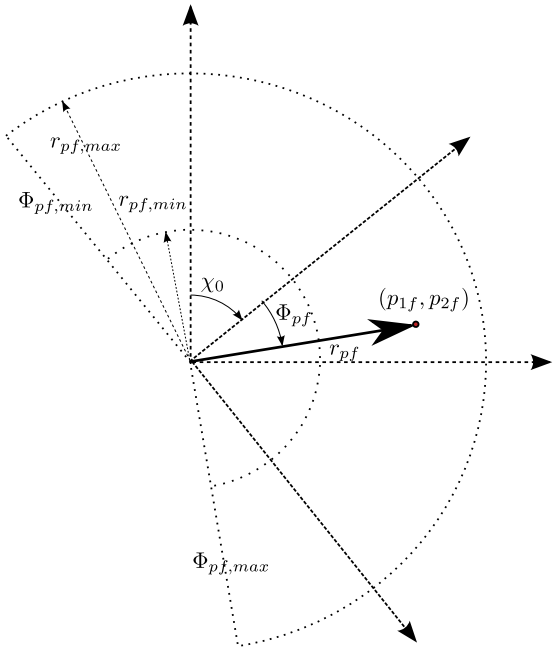



Figure 5.5: Illustration of the final position's coordinate transformation of and limit imposition.

mization variable becomes $p_{3f}$, $\Phi_{pf}$ and $r_{pf}$:

$$p_{1f} = r_{pf}\cos\left(\Phi_{pf} + \chi_0\right), \quad p_{2f} = r_{pf}\sin\left(\Phi_{pf} + \chi_0\right), \tag{5.4.4}$$

and subject to:

$$\Phi_{pf,max} \geq \Phi_{pf} \geq \Phi_{pf,min}, \quad V_{max}t_f \geq r_{pf} \geq V_{min}\frac{t_f}{10}, \quad \dot{p}_{3max}^{CR}t_f \geq p_{3f} \geq \dot{p}_{3min}^{CR}t_f. \tag{5.4.5}$$

where $\Phi_{pf,max}$ and $\Phi_{pf,min}$ are the design parameters to control the limit of the

azimuth of the final position.

- $5^{\text{nd}}$ – $10^{\text{th}}$ OVs: the initial and final jerks of the trajectory are subject to the following bound constraint:

$$\dddot{p}_{max} \geq \dddot{p}_{i0} \geq -\dddot{p}_{max} \quad \text{and} \quad \dddot{p}_{max} \geq \dddot{p}_{if} \geq -\dddot{p}_{max}, i = 1,2,3, \tag{5.4.6}$$

  with $\dddot{p}_{max}$ as a design parameter.

- $11^{\text{nd}}$ – $13^{\text{th}}$ OVs: the final acceleration is subject to:

$$a_{max}^{long} \leq \left|\ddot{p}_{1f}\right|, \quad a_{max}^{long} \leq \left|\ddot{p}_{2f}\right|, \quad a_{max}^{norm} \leq \left|\ddot{p}_{3f}\right|. \tag{5.4.7}$$

  where $a_{max}^{long}$ and $a_{max}^{norm}$ are the limits on the longitudinal and normal accelerations, obtained from the aircraft performance model.

- $14^{\text{nd}}$ OV: final flight path angle $\gamma_f$ is subject to:

$$\arcsin\left(\frac{\dot{p}_{3,min}}{V_{min}}\right) \geq \gamma_f \geq \arcsin\left(\frac{\dot{p}_{3,max}}{V_{min}}\right). \tag{5.4.8}$$

- $15^{\text{nd}}$ OV: final ground track $\chi_f$ is subject to:

$$\chi_0 + \Delta\chi \geq \chi_f \geq \chi_0 - \Delta\chi, \tag{5.4.9}$$

  with $\Delta\chi$ denotes the maximum ground track deviation (design parameter) and $\chi_0$ denotes the initial ground track.

**Initial Guess** Considering the above feasible solution space and a bias to the resolution of turning right, the following initial guess is chosen as the start search point for the subsequent constraint violation minimization process:

$$\boldsymbol{\Xi}_0 = \begin{bmatrix} \Xi_1 & p_{1f} & p_{2f} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_f & \chi_f \end{bmatrix}^T. \tag{5.4.10}$$

with

$$\Xi_1 = \left(V_f{}'_{max} - V_{f\,min}\right)/2 + V_{f\,min} \quad \text{and} \quad p_{if} = \frac{p_{i\,max}}{2}, \quad i = 1,2.$$

### 5.4.2 Problem Scaling

Scaling affects everything, as warned by Betts [2010]:

> *Poor scaling can make a good algorithm bad. Scaling changes the convergence rate, termination tests, and numerical conditioning.*

**Optimization Variables** The OVs with upper and lower bounds are scaled using:

$$\bar{\Xi}_i = \frac{2(\Xi_i - \Xi_{imax})}{\Xi_{imax} - \Xi_{imin}} + 1, \tag{5.4.11}$$

so that the scaled OVs are in the range $\bar{\Xi}_i \in [-1, 1]$.

Therefore, when setting up the optimizer, the OVs' bounds and step lengths of the search direction should be selected accordingly.

**Constraint Violations** The constraint vector $\boldsymbol{c}$ ($\boldsymbol{\Xi}$) is normalized such that, when the constraint is violated, its elements have a similar magnitude, i.e. $\bar{c}_i \approx 1$.

For bounded constraints, the following *relative constraint violation* is used, when a maximum constraint is imposed:

$$\bar{c} = \begin{cases} \frac{x - x_{max}}{|x_{max}|} & \text{if } x_{max} \neq 0, \\ x & \text{otherwise.} \end{cases}, \tag{5.4.12}$$

and when a minimum constraint is imposed:

$$\bar{c} = \begin{cases} \frac{x_{min} - x}{|x_{min}|} & \text{if } x_{min} \neq 0, \\ -x & \text{otherwise.} \end{cases}. \tag{5.4.13}$$

As a result, although the elements of the constraint vector represent different types of constraint with different limit ranges, e.g. $V \in [78, 120]$ m/s and $\mu \in [-0.785, 0.785]$ rad, the relative constraint violation, expressed as the ratio of the constraint violation to its extreme value, can be used to judge the level of violation regardless of the magnitude of the extreme value.

*Remark* 5.4. Note that, as will be seen in §5.4.4, the constraints will be handled by a *squared 2-norm penalty function* and a major characteristic of the squared 2-norm

penalty function is that it places extreme emphasis on constraint violations larger than one and little emphasis on violations less than one. Therefore, the normalized constraint violations will be multiplied by 100 to express the relative constraint violation with percentage.

**Cost Functions** The cost function $F(\boldsymbol{\Xi})$ in the NLP problem is scaled using the weighted sum of the normalized cost components:

$$\bar{F} = \sum_{i=1}^{N_J} w_i \bar{F}_i, \tag{5.4.14}$$

with the normalized cost components $F_i \approx 1$:

$$\bar{F}_i = \left| \frac{F_i - F_{i,/,des}}{F_{i,/,des}} \right| \tag{5.4.15}$$

where $N_J$ is the number of cost components, $w_i \in [0,1]$ is the relative importance, $F_{i,des}$ is the desired value of the cost component.

Hereafter, the scaled optimization variables, constraint functions, and cost functions will be used, although the over bar accent $\bar{\cdot}$ is dropped for writing convenience and consistency with the notation in literatures.

### 5.4.3 Penalty Function

Problem 5.3.1 can be recast into the following unconstrained NLP problem with the *penalty method* [Griffin & Kolda, 2010]:

**Problem 5.4.1** (Nonlinear Programming Problem with Penalty Function)**.** *Find*

$$\boldsymbol{\Xi}^* = \underset{\boldsymbol{\Xi} \in \mathbb{S}}{\arg\min}\, F(\boldsymbol{\Xi}) + \mathcal{P}\left(\boldsymbol{c}^+(\boldsymbol{\Xi}), \rho_k\right), \tag{5.4.16}$$

*with* $\mathcal{P}$ *as a* penalty function*:*

$$\lim_{\rho_k \to \infty} \mathcal{P}\left(\boldsymbol{c}^+(\boldsymbol{\Xi}), \rho\right) = \begin{cases} +\infty & \text{if } \forall c_i^+ \in \boldsymbol{c}^+, \quad \exists c_i^+ > 0, \\ 0 & \text{otherwise}. \end{cases} \tag{5.4.17}$$

*to penalize the constraint violations* $\boldsymbol{c}^+$*:*

$$c_i^+ (\boldsymbol{\Xi}) = \max(0, c_i(\boldsymbol{\Xi})), \tag{5.4.18}$$

*where* $\rho$ *is referred as the* penalty parameter *and determines the severity of the penalty.*

Among other penalty function options proposed in Griffin & Kolda [2007], the squared 2-norm penalty function:

$$\mathcal{P}_{\ell_2^2} \left( \boldsymbol{c}^+ (\boldsymbol{\Xi}) , \rho \right) = \rho ||\boldsymbol{c}^+ (\boldsymbol{\Xi}) ||_2^2 \tag{5.4.19}$$

is selected after some preliminary tests. Figure 5.6 shows the contour plot of the penalty functions as the constraint violations are varying from 0 to 100, illustrating how the constraint violation will be penalized. This penalty function will be used to handle constraint in the next section.

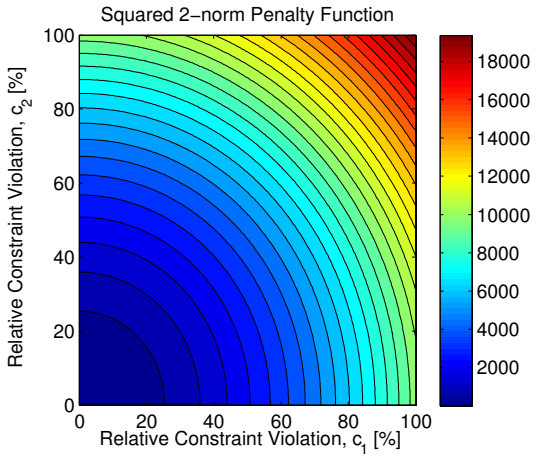



Figure 5.6: Squared 2-norm penalty function contour plot with a two-dimensional constraint vector.

### 5.4.4 Optimization

**Nonlinear Programming Algorithms** Although there exists numerous solvers for the transcribed NLP problems, it is found that the gradient-based solvers (alone), such as the SNOPT solver Gill *et al.* [2008] and the MATLAB `fmincon` function, are not as

effective as the derivative-free solvers, such as the Hooke-Jeeves pattern search, which has been mentioned by Yakimenko [2000], and further shown by Drury [2010] and by Lai & Whidborne [2010].

A derivative-free solver implementing the pattern search algorithm by Hooke & Jeeves [1961, as cited in [Kelley, 1999]] is used in this work. The solver is based on the MATLAB implementation by Kelley [1999, §8.3]. As can be seen in Algorithm-C.2, the Hooke-Jeeves algorithms only involves two basic operations: *exploratory move* and *pattern move* and is relatively easy to implement. The simplicity of this algorithm makes it a suitable candidate for on-board implementation. The algorithm parameters are summarized in Table C.1.

**Two-stage Approach** A two-stage approach, as suggested by Yakimenko [2000] and tested by the author in Lai & Whidborne [2010], is implemented.

In the first stage, with the purpose to achieve a feasible point, the initial guess $\boldsymbol{\Xi}_0$ from §5.4.1 is used as the starting point for to the Hooke-Jeeves (HJ) algorithm to minimize the constraint violations, that is to solve:

**Problem 5.4.2** (Constraint Violations Minization)**.** *Find*

$$\boldsymbol{\Xi}_1 = \underset{\boldsymbol{\Xi}\in\mathbb{S}}{\arg\min} \, ||\boldsymbol{c}^+(\boldsymbol{\Xi})||_2^2. \tag{5.4.20}$$

The solutions $\boldsymbol{\Xi}_1$ are normally feasible in our tests, otherwise the resulting relative constraint violations are mostly within the range of 1 to 20 %.

In the second stage, the $\boldsymbol{\Xi}_1$ is used as the starting point for the HJ algorithm to minimize the cost and the penalty function, that is to solve:

**Problem 5.4.3** (Cost Optimization)**.** *Find*

$$\boldsymbol{\Xi}_2 = \underset{\boldsymbol{\Xi}\in\mathbb{S}}{\arg\min} \left( F(\boldsymbol{\Xi}) + \rho||\kappa\, \boldsymbol{c}^+(\boldsymbol{\Xi})||_2^2 \right), \tag{5.4.21}$$

*where $\rho$ is set to a large number to prevent any reduction of the existing feasibility, the scale factor $\kappa$ is used to scale the magnitude of the left-over constraint violations from $\approx 1$ to about $\approx 100$ so that the 2-norm penalty function can be more effective.*

With a limited budget of function evaluations, the solutions $\boldsymbol{\Xi}_2$ are normally sub-optimal but mostly feasible in our tests (about 88.79% of the test cases, see §5.5.2).

## 5.5 Main Results

This section presents the testing results of the developed trajectory planning algorithm. The experiment set up is first introduced and then the experiment results are organized in the subsequent two subsections to:

1. investigate the method's ability to generate feasible and good trajectories in real time, by looking into the success rate, optimality and computational load over all test cases;

2. investigate the usability of the resulting trajectories when no feasible solutions can be found by the solver in the tightly constrained or initially infeasible situation. This is done by looking into the constraint violation of the resulting trajectories.

### 5.5.1 Experiment Settings

**Testing Scenarios** The testing encounters are obtained from the previous simulation experiment in §4.4. There are, in total, 1040 encounters in which an alerted conflict still resulted in an NMAC. For each simulated encounter, the navigation state, tracker state, and route information (at the moment when the conflict is first issued) are extracted and used as the initial condition of a testing scenario for the subsequent experiments. The actual initial linear time-to-CPA $t_{\mathsf{C}_{lin}}$, Vertical Miss Distance (VMD) and HMD are also calculated. For analysis purpose, an encounter is regarded as *critical* when:

$$t_{\mathsf{C}_{lin}} < 3\,\mathrm{s} \quad \text{or} \quad (hmd < 710\,\mathrm{ft} \wedge vmd < 140\,\mathrm{ft})\,. \tag{5.5.1}$$

Considering the definition of an NMAC (a 500 ft and 100 ft cylinder) and the introduced robustness margin (5.2.9) (i.e. at least a factor of $\sqrt{2}$), a critical situation would normally be unresolvable for the trajectory planning problem, from the perspective of obstacle constraints.[1] There exist 63 out of 1040 testing encounters (about 6%) that are in critical situations.

Furthermore, there are 318 testing encounters (about 36%) that are initially infeasible, according to the initial navigation state and the aircraft performance model

[1]Note that, there exist 9 out of 63 critical encounters that are still solvable in our experiment.

used for trajectory planning (whose performance is slightly reduced from that of the simulation model to leave some performance margin for trajectory tracking). These initial infeasibilities on performance constraints are caused by the reduced aircraft performance and the model approximation (introduced as assumptions in §5.2.1). As in the real world, the model mismatch and, sometimes the initial infeasibilities, are inevitable in practical applications. Some of these initial infeasibility can be recovered by the algorithm, which will be discussed further in §5.5.3.

In §5.5.2, to purely measure the trajectory planning method's ability to generate feasible trajectories, the initial infeasibility will be removed by modifying the navigation states to satisfy the performance constraints. Therefore, the are 1040 testing encounters, with 63 critical encounters and no initial infeasibility.

**Parameters Settings** All the parameters used in the simulation experiments are summarized in the aircraft performance data in Table B.1 and the trajectory planning algorithm parameters in Table C.1.

### 5.5.2 Result Statistics

**Success Rate** The ability to generate feasible trajectories is measured by the percentage of successful testing encounters to total testing encounters. Success was defined as satisfying the specified constraint violation tolerance within the specified budget of maximum function evaluations. Considering the conservative aircraft performance model and the virtually enlarged obstacle protected volume, a solution trajectory with 1% of the relative constraint violations (with respect to its limit values) is regarded as *feasible*.

Table 5.1 summarizes the success, failure and critical situation rates. As can be seen, the algorithm (with a budget of 4500 maximum function evaluations) is able to find a feasible trajectory in 88% of the testing encounters. Besides the 6.06% of critical situations, there are still 6.15% encounters in which no feasible solution can be found by the algorithm. The usability of the resulting trajectories of the infeasible solutions will be further investigated in §5.5.3.

Table 5.1: Success rate for the testing of the trajectory planning algorithm over 1040 test cases.

| Type | Count | Percentage [%] | Percentage [%] (Excluding Critical) |
|---|---|---|---|
| Success | 913 | 88.79 | 93.45 |
| Failure | 64 | 6.15 | 6.55 |
| Critical Situations | 63 | 6.06 | - |

**Optimality** The ability to generate good (operational suitable) trajectories is measured by the relative optimality. The relative optimality is used to show that the optimization in the algorithm is taking effect to bring down the specified operational cost. To obtain the relative optimality, the algorithm was run twice: with and without the cost function; and then, the cost values of all solution trajectories generated by the algorithm with the cost function were summarized and served as the baseline (pseudo-optimal) for comparison. Relative optimality is defined as the ratio of the cost difference in two runs to the pseudo-optimal cost value. The larger the relative optimality, the bigger the difference between two runs, and thus the better the algorithm's ability to minimize the operational cost.

Figure 5.7 shows the relative optimality and the mean cost values of two runs over 1040 test cases, in which the overall cost were decomposed into eight components[1] to enable the component-wise comparison. It can be seen from the figure that:

(a) the pseudo-optimal cost is one third smaller than the one without cost minimization, showing the effectiveness of the optimization process in the algorithm; the relative optimality is 0.47, which on its own can tell little but this measure will be used in §6.3 for more comparison;

(b) in both situations, the components' average cost values appear uneven, indicating the optimization process may have focused on some particular components;

(c) the relative optimality of components 4 to 8 show better performance than others, as they are related to the optimization variables directly; furthermore, while

[1]Note that, the final costs function used in this experiment are the final flight path angle deviation, ground track deviation and acceleration, which is equivalent to the final cost specified in (5.2.20).

the route deviation and intruder invisibility cost have been reduced, the converging duration/magnitude cost remained at a similar level even when the cost minimization is enabled.

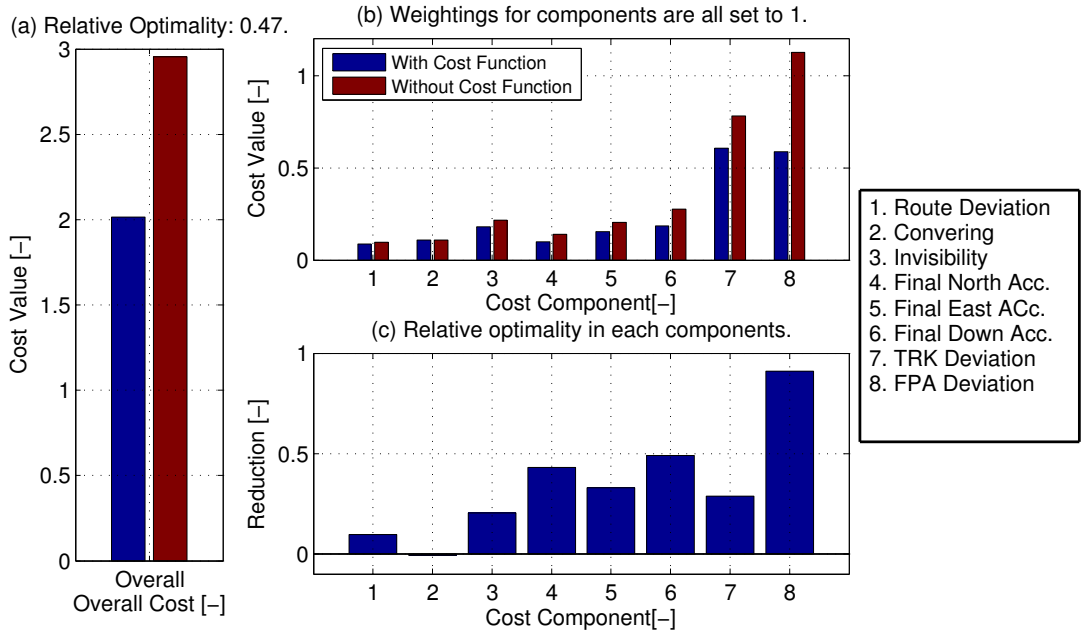



Figure 5.7: Average cost values of two test runs and their relative optimality.

**Computational Load** The algorithm was implemented in MATLAB R2013a and the simulation was performed on a 64-bit OS X operating system with a 2.7 GHz Intel Core i7 processor. The computational load is measured by the number of function evaluations invoked by the NLP algorithm and the computation time obtained with the `tic` and `toc` functions in MATLAB.

Table 5.2 summarizes the average computational load over all testing scenarios. It can be seen that it took an average of 2.56 s to obtain a solution, and 80% of which (2.07 s) was spent on the cost minimization process. The variation of the computation time is illustrated with its histogram in Figure 5.8a, from which it can be seen that the maximum computation time is less than 5s.

Regarding the number of function evaluations, it took an average of 750 evaluations (25% of its limit) in the first phase to obtain a feasible solution. However, in the second

Table 5.2: Summary of the average computational load for the trajectory planning algorithm over 1024 testing scenarios.

| | Constraint Satisfaction | Cost Minimization | Total |
|---|---|---|---|
| Function Evaluation [-] | 750 | 1462 | 2212 |
| Computation Time [s] | 0.49 | 2.07 | 2.56 |
| Time per Evaluation [s] | $7.69 \times 10^{-4}$ | $4.09 \times 10^{-3}$ | $1.24 \times 10^{-3}$ |

phase, it spent an average of 1462 evaluation (i.e. 97% of its limit), which means the cost minimization process was normally stopped due to the function evaluation limit.

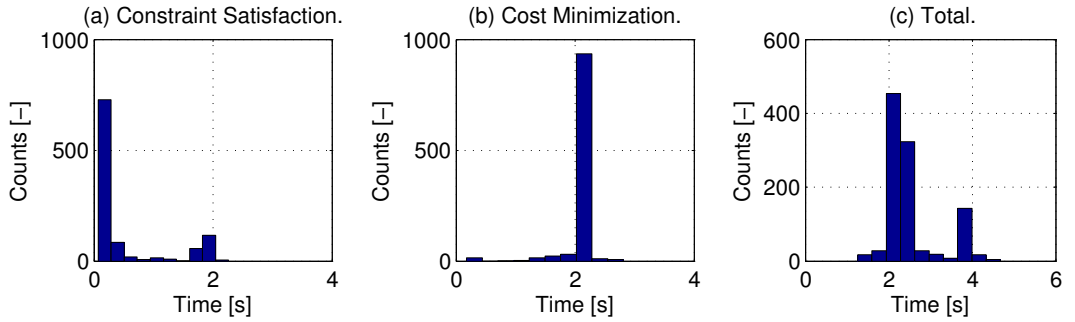
Lastly, as can be seen from the *time per evaluation* measures, each evaluation in the cost minimization process (including both cost and constraint function evaluation) is one order of magnitude more expensive than that in the constraint satisfaction process (include the constraint function evaluation only), which means the cost function evaluation is the most expensive part to compute in this implementation.

### 5.5.3 Example Trajectories

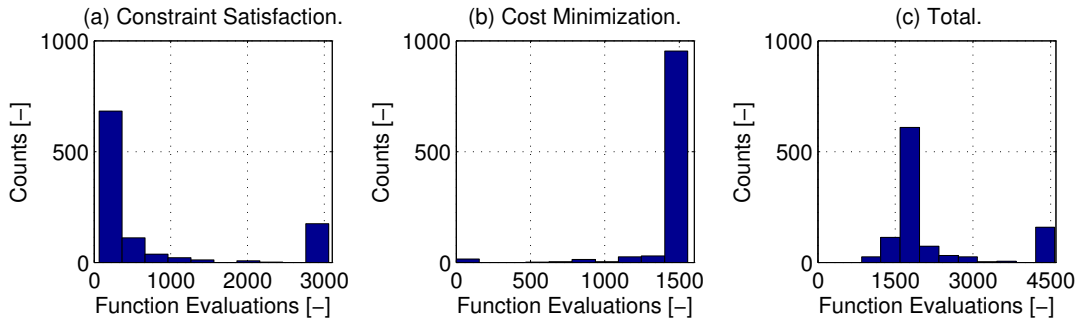
**Infeasible Trajectory Characteristics** For analysis purposes, Table 5.3 categorizes all the result trajectories into five types according to their constraint violations: i.e. feasible, acceptable, infeasible-obstacle, infeasible-performance, and both-infeasible trajectory. Figure 5.9 shows four histograms of the maximum constraint violations of these trajectories.[1] A *typical* trajectory of each type is then chosen for demonstration. A typical trajectory of each type is an example trajectory with its maximum constraint violations at the level that appears most frequently in Figure 5.9.

Figures C.2-C.5 in Appendix C show, respectively, the examples of acceptable, infeasible-obstacle, infeasible-performance, and both-infeasible trajectories. In each figure, the resulting miss distance and constraint violations of the each infeasible-solution trajectory are highlighted. Although no feasible solutions could be found, within the maximum number of function evaluations, for these situations, it can be seen

[1]Note that, there are 16 unresolvable scenarios, in which both the host and intruder were initialized at the same position.




(a) Computation Time.




(b) Function Evaluation.

Figure 5.8: Histogram of the computational load required by the trajectory planning algorithm with a budget of 4500 function evaluation.

Table 5.3: Percentage of five types of result trajectories, categorized according to their constraint violations.

| Type | Definition | Percentage [%] | Example |
|---|---|---|---|
| Feasible | $\|\|\boldsymbol{c^+}\|\|_\infty \leq \boldsymbol{1}$ | 88.79 | Fig. C.1 |
| Acceptable | $\boldsymbol{1} \leq \|\|\boldsymbol{c^+}\|\|_\infty \leq \boldsymbol{10}$ | 2.60 | Fig. C.2 |
| Infeasible Obstacle | $(c_o^+ \geq \boldsymbol{10}) \wedge \left(\|\|\boldsymbol{c^+}_p\|\|_\infty \leq \boldsymbol{10}\right)$ | 5.96 | Fig. C.3 |
| Infeasible Performance | $(c_o^+ \leq \boldsymbol{10}) \wedge \left(\|\|\boldsymbol{c^+}_p\|\|_\infty \geq \boldsymbol{10}\right)$ | 1.06 | Fig. C.4 |
| Both Infeasible | $(c_o^+ \geq \boldsymbol{10}) \wedge \left(\|\|\boldsymbol{c^+}_p\|\|_\infty \geq \boldsymbol{10}\right)$ | 2.60 | Fig. C.5 |

that the aircraft performance required by these resulting trajectories' are very close to the specified limit and their miss distances are all larger than the required separation.

**Initial Infeasibility Recovery** As discussed in §5.5.1, there exist 318 testing encounters that are initially infeasible, due to the violation of the performance constraint

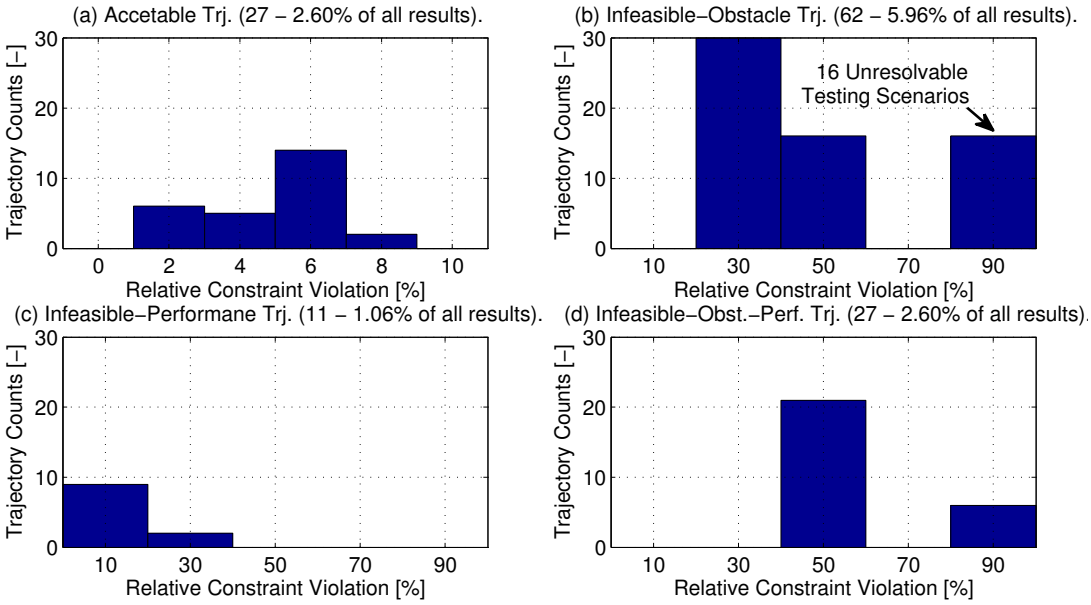



Figure 5.9: Maximum constraint violations summary over all infeasible trajectory results.

according to the reduced-performance model. The type and magnitude of the initial infeasibility (violated constraints) are summarized in Figure 5.10. It can be seen from the constraint type figure, that initial throttle violations account for the majority of the initial infeasibility, and the remaining are the airspeed and vertical rate only.

In the experiments, when the violation magnitude is small (about at the level of 25%), the initial infeasibility can be recovered by the algorithm in the next time step. Figure 5.11 shows an example of recovering from an initial infeasible throttle setting. In case of the 318 initially infeasible cases, 197 of them (62%) were recovered in the next time steps while some others may take more steps.

*Remark* 5.5. The large number of initially infeasible throttle shows the mismatch of the simulation model and the reduced-order model used for trajectory planning. Although this mismatch is inevitable, further investigation is required to study its effect on the trajectory tracking performance.

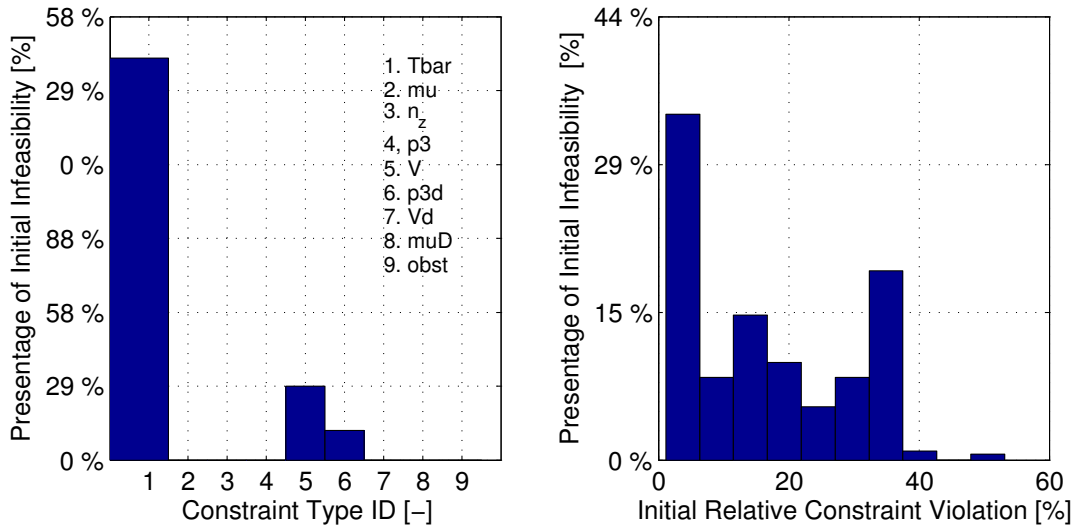


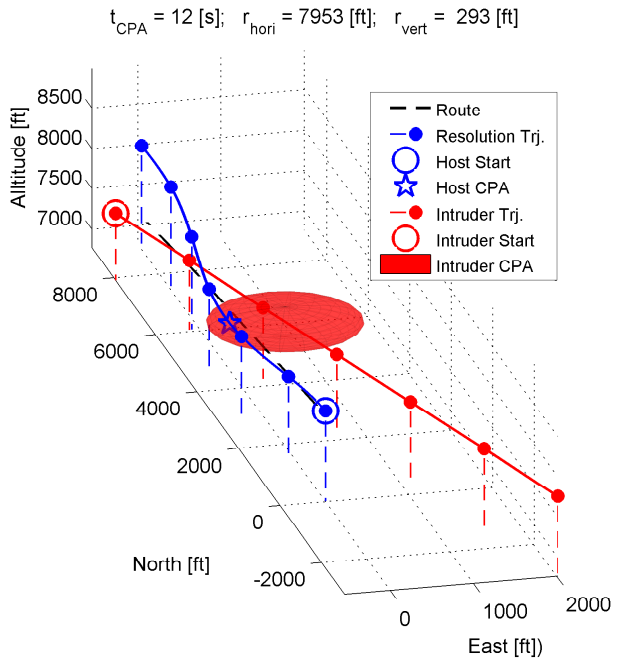

Figure 5.10: Statistics of the initial infeasibility.

## 5.6 Summary

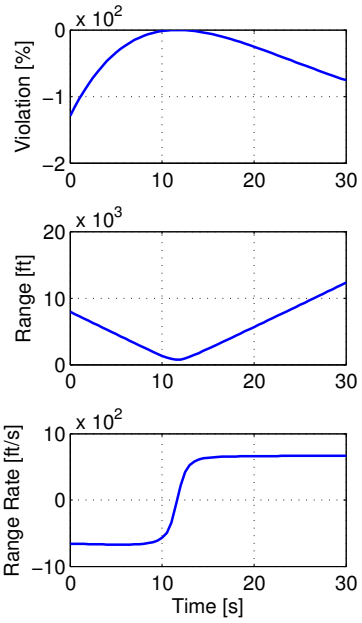
This chapter has formulated a trajectory planning problem to incorporate the requirements for conflict resolutions manoeuvres and then presented the development of a trajectory planning algorithm to solve the problem. The algorithm has been tested with 1040 encounters, in which an alerted conflict still resulted in an NMAC in the experiment in §4.

The main testing results showed that the algorithm is able to find a feasible trajectory in 88% of the testing encounters, using 2.56 s computation time and 2212 function evaluations in average. With the function evaluation budget of 4500, the maximum computation time over all cases is 5 s. The operational suitability of the generated trajectories has been investigated with a relative optimality measure and the results show that the algorithm is able to bring down one third of the original (without cost minimization) operational cost.

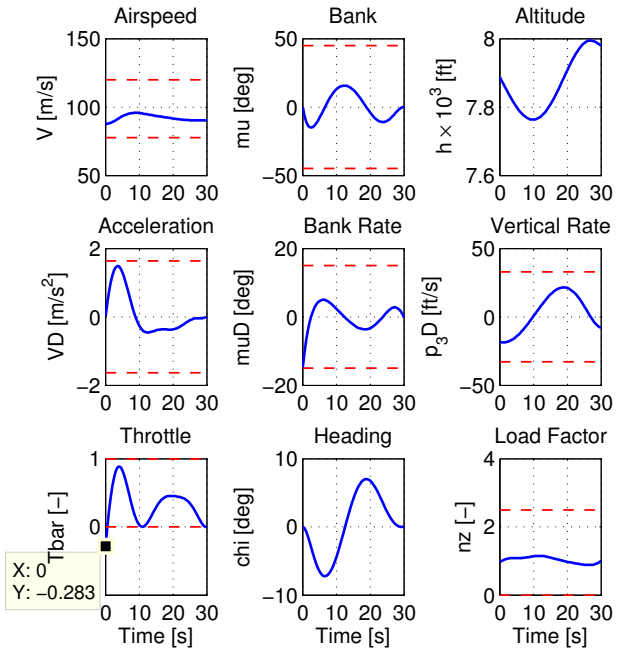
Furthermore, closer inspection has been carried out with the initially infeasible cases and the failure cases, in which the algorithm failed to find a feasible solution within the given budget. The typical trajectory examples indicated that 1) the algorithm is able to recover most (about 62%) of the initial infeasibility in the second time



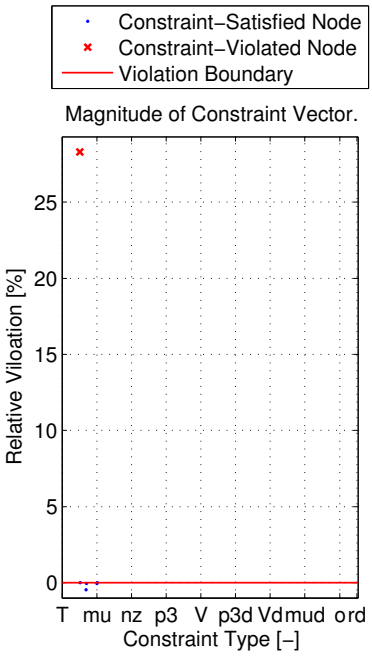

(a) Resolution trajectory and the initial linear time-to-CPA, horizontal and vertical range.




(b) Obstacle constraint and relative range.




(c) Aircraft state and control trajectories and their performance limits.




(d) Constraint vector evaluated at all the collocation points (nodes).

Figure 5.11: An example trajectory showing the recovery from an initial infeasibility. Although the initial throttle of the reduced-order model is −0.283, the subsequent throttles fall back into the feasible region as soon as the second time step.

step; and 2) despite the infeasibility of solutions, the resulting trajectories' required performance are very close to the specified limit and their miss distances are normally larger than the required separation.

The closed-loop performance of the algorithm and the trajectory usability of the infeasible solutions will be further investigated in the next chapter.

This page intentionally contains only this sentence.

# Chapter 6

# System Integration and Performance Evaluation

## 6.1 Introduction

As shown in the previous chapter, the trajectory planning process can take up to 5 seconds. For an aircraft travelling at 100 m/s, this is a relatively long time and will inevitably introduce a time latency into the system. While §6.2.1 shows how a low-level implementation can further reduce the computation time, §6.2 describes a three-layer architecture (see Gat [1998]) that is used to explicitly handle the time latency and to integrate the planning process into the collision avoidance logic.

Although the final product of §6.2 is a proof-of-concept software prototype of the trajectory-planning collision avoidance logic, it is worth noting that, this initial prototype does not implement a mechanism to detect conflicts (or to delay the issue of a conflict alert) based on the resolution trajectory from the planning process. Therefore, for the initial feasibility study in this thesis, this chapter regards the prototype as a *conflict resolution logic*, and simply integrates it into the baseline collision avoidance logic by replacing the original geometric conflict resolution logic. Using the same conflict detection mechanism in both candidates can facilitate a direct comparison of the conflict resolution capabilities between the geometric (reactive) and trajectory-planning (deliberative) methods.

In order to study the feasibility and potential benefits of the trajectory-planning-

based collision avoidance logic, the prototype was tested in three simulation experiments with the objectives of 1) safety performance evaluation, 2) robustness analysis to sensor noise, and 3) investigation of the FOR effect; and the results were compared with those of the baseline performance described in §4.4.2. Section 6.3 describes the experiment settings and presents these comparisons.

Finally, a closer inspection of four observed phenomena—ineffectiveness of the emergency strategy, conservativeness in uncertainty propagation, parallel-track encounters and field-of-regard effect—are discussed and demonstrated with examples in §6.4.

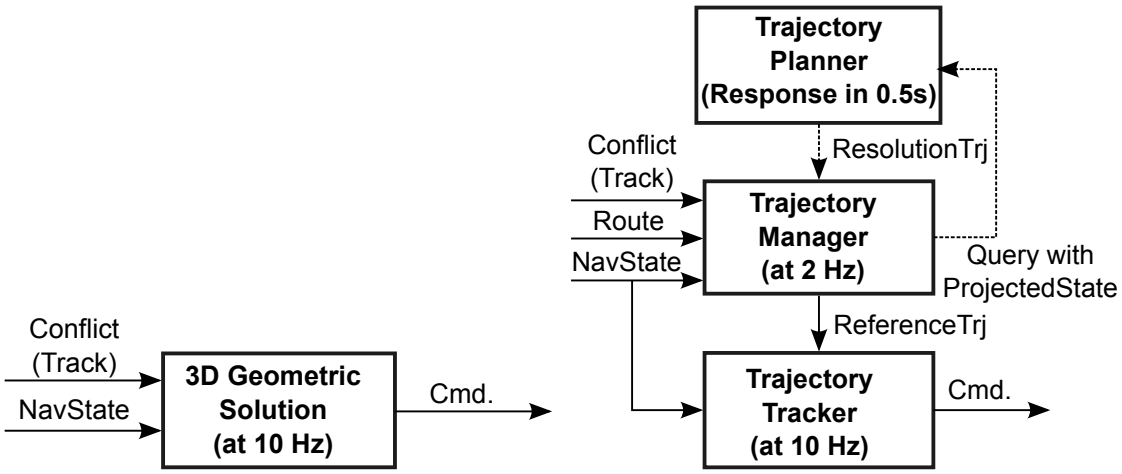
## 6.2 Integration of Trajectory Planning

Figure 6.1 shows the architectures of two conflict resolution logics. The *reactive geometry-based logic*, shown in Figure 6.1a, uses the conflict resolution logic described in §3.6, which can be updated in real time as fast as 10 Hz. The *deliberative trajectory-based logic*, shown in Figure 6.1b, uses the three-layer architecture to integrate the planning process of trajectory planning into the collision avoidance system. Both conflict resolution logics are interfaced with the collision avoidance system described in Figure 3.18. The three components of the three-layer architecture will be described in detail in the following subsections.

### 6.2.1 Trajectory Planner

**System Description** Figure 6.2 shows the concept of operation and the interface of the Trajectory Planner, which takes as input the projected states (navigation, tracker and route states) from the Trajectory Manager and computes a resolution trajectory as output, using the algorithm described in §5.4. The Trajectory Planner is able to answer the query from the Trajectory Manager in a response time of 0.5 s in our implementation.

**Real-time Performance** The computation time of the Trajectory Generation Algorithm (TGA) can be further reduced by generating the MATLAB Executable (MEX) file from the MATLAB source code. In order to verify the MEX implementation




(a) Reactive geometry-based. (b) Deliberative trajectory-based.

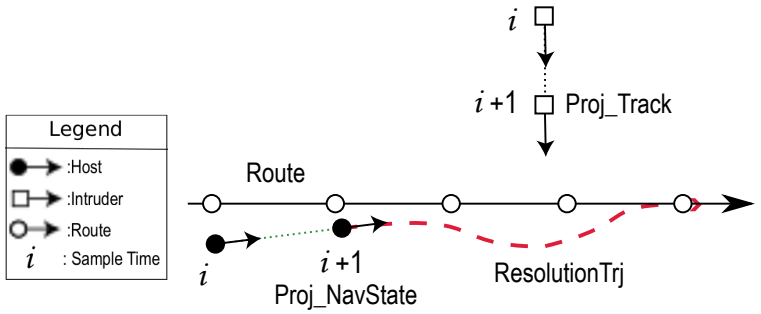
Figure 6.1: Architecture of two conflict resolution functions.

against the MATLAB implementation, the two implementations (i.e. TGA and TGA-MEX) were tested with the 1040 scenarios in §5.5.1.

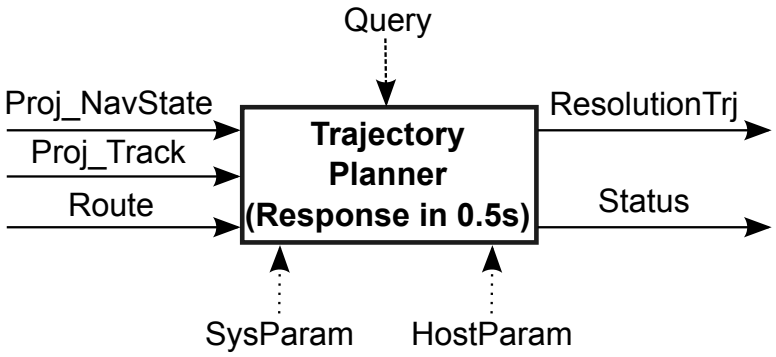
Figure 6.3 summarizes the resulting computation time, number of function evaluation and cost function with nine histograms. Compared with those of the MATLAB implementation, the MEX implementation is about ten times faster and its computation time is reduced to 0.5 s at its maximum. Moreover, the function evaluations and the cost function of both implementations show exactly the same distributions, verifying that both implementations had used the same number of function evaluations to get the same results.

### 6.2.2 Trajectory Manager

In order to integrate the deliberative process into the collision avoidance system, the Trajectory Manager (as the sequencer in the three-layer architecture) is required to handle the issues, such as time alignments and coordinate transformations, due to the non-zero computation time. The main function of the Trajectory Manager is to maintain a *resolution plan*, so as to provide the Trajectory Planner with the projected states to perform the planning, and to provide the Trajectory Tracker with an *execution trajectory* to facilitate a correct execution of the resolution trajectory and to ensure the accuracy of the projected state.




(a) Concept of operation.




(b) Interface.

Figure 6.2: The system description of the Trajectory Planner.

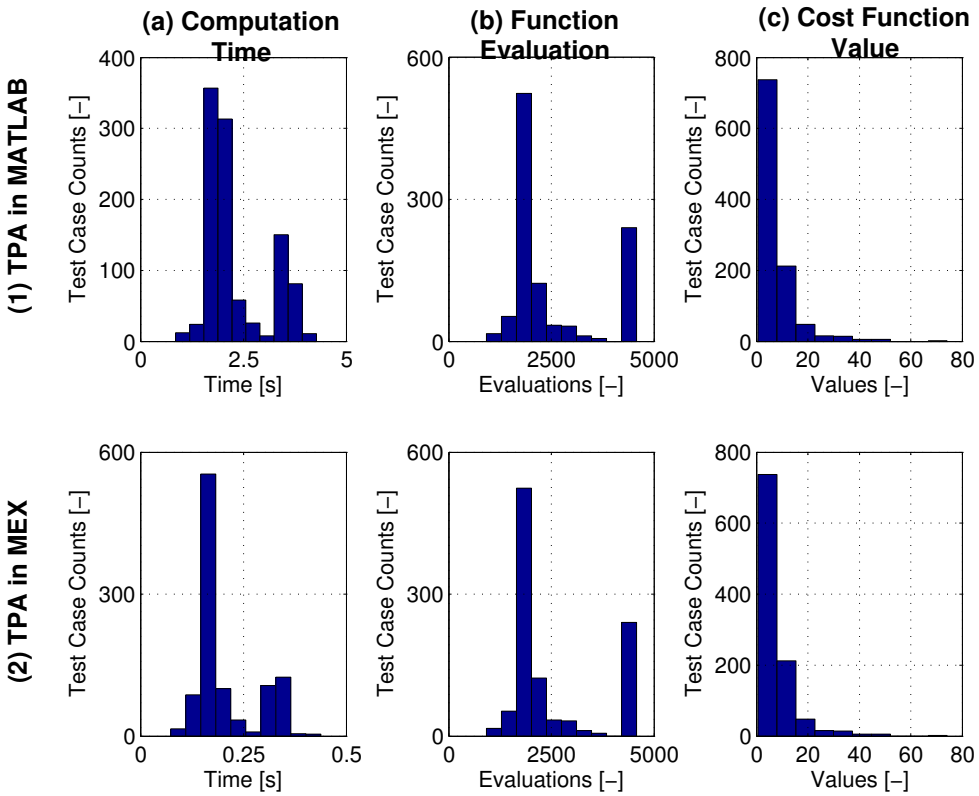



Figure 6.3: Result histograms of the (a) computation time, (b) number of function evaluations and (c) cost function values of three different implementations of the Trajectory Planning Algorithm (TPA): (1) TPA in MATLAB and (2) TPA in MATLAB executable (MEX).

A resolution plan, consisting of an execution trajectory, a reference trajectory and a projected state, is depicted in Figure 6.5. Also illustrated in Figure 6.5 are the three main functions used to update the resolution plan: *plan generation*, *plan assessment* and *plan selection*. The interfaces and data flow among these functions are shown in Figure 6.4.

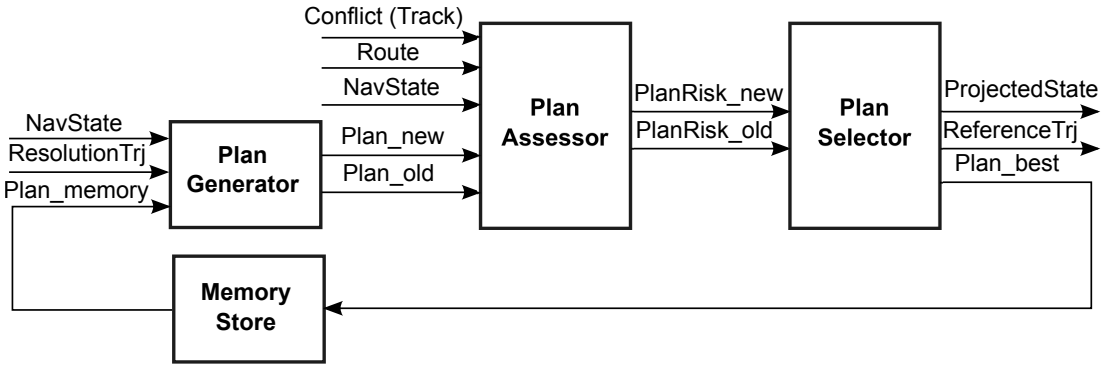



Figure 6.4: Trajectory Manager architecture, consisting of Plan Generator, Plan Assessor and Plan Selector.

In particular, for the results presented in this work, the *criteria* used by the Plan Assessor to assess the risk and performance of each resolution plan are as followed:

- Direct Collision Risk: a logical value indicating whether the initial state of a resolution plan satisfies the collision condition as specified in (3.6.10);
- Trajectory Collision Risk: the maximum value of the obstacle constraint violation $\boldsymbol{\phi}_o$ in (5.2.13) over the whole plan;
- Trajectory Flyability Risk: the maximum value of the performance constraint violation $\boldsymbol{\phi}_c$, $\boldsymbol{\phi}_x$ and $\boldsymbol{\phi}_d$ in (5.2.13) over the whole plan;
- Trajectory Deviation Risk: the Euclidean distance between the current aircraft position and the initial position of the resolution plan.

The *metrics* used in Figure 6.5 is the weighted sum of the above criteria and will be used by the Plan Selector to make its decision about the better plan.

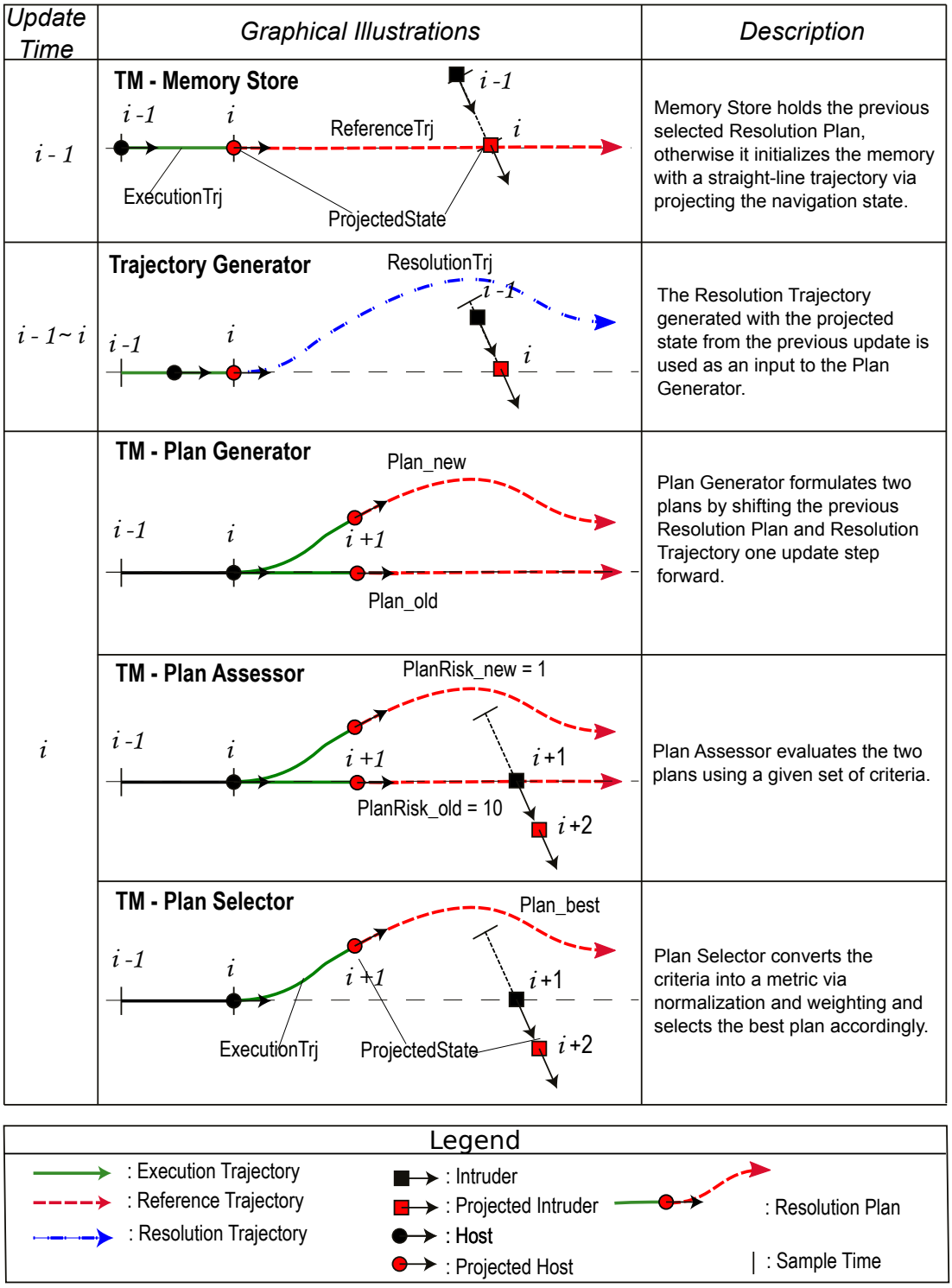



Figure 6.5: The Trajectory Manager's concept of operation.

### 6.2.3 Trajectory Tracker

The Trajectory Tracker takes as input the execution trajectory selected by the Trajectory Manager and outputs the autopilot command required to track this trajectory. Although the execution trajectory consists of the host's desired position and velocity trajectories, only the position trajectory is required to be tracked accurately for a successful resolution.

Figure 6.6 depicts the notation used for trajectory tracking, in which:

- $\mathtt{H}$ denotes the host aircraft and $\mathtt{D}$ denotes the current desired point on the execution trajectory;
- $\boldsymbol{p}$, $\boldsymbol{p}$, and $\boldsymbol{e}$ denote the position, velocity and error vectors;
- $\mathsf{R}$ denotes the resolution coordinate system and $\mathsf{D}$ denotes a *desired path coordinate system* that has its origin at the current desired point and its x-direction and z-direction aligned with the desired horizontal velocity and the downward direction.

The architecture of the Trajectory Tracker is based on the notion of *two degree of freedom controller design* (see Murray [2010, §1.1]). This is a standard technique in linear control theory that separates a controller into a feedfoward compensator and a feedback compensator, as shown schematically in Figure 6.7.

Firstly, the feedforward compensator generates the current desired (nominal) control and state signals, according to the execution trajectory. As the execution trajectory is generated in the resolution coordinate system $\mathsf{R}$ and probably at an earlier time (as shown in Figure 6.6), the navigation state needs to be transformed to the resolution coordinate system and the execution trajectory needs to be interpolated according to the current time. This gives the desired control $\boldsymbol{u}_d$, desired state $\boldsymbol{x}_d$ and actual state $\boldsymbol{x}_a$:

$$\boldsymbol{u}_d = \begin{bmatrix} V_H \\ \chi \\ V_V \end{bmatrix}_d, \quad \boldsymbol{x}_d = \begin{bmatrix} \boldsymbol{p}_{\mathtt{D}} \\ \boldsymbol{v}_{\mathtt{D}} \end{bmatrix}, \quad \boldsymbol{x}_a = \begin{bmatrix} \boldsymbol{p}_{\mathtt{H}} \\ \boldsymbol{v}_{\mathtt{H}} \end{bmatrix}. \tag{6.2.1}$$

where $V_{H,d}$, $V_{V,d}$, and $\chi_d$ are the cylindrical coordinates of the current desired inertial velocity; $\boldsymbol{p}_{\mathtt{H}}$ ($\boldsymbol{v}_{\mathtt{H}}$) and $\boldsymbol{p}_{\mathtt{D}}$ ($\boldsymbol{v}_{\mathtt{D}}$) are the current actual and desired position (velocity)

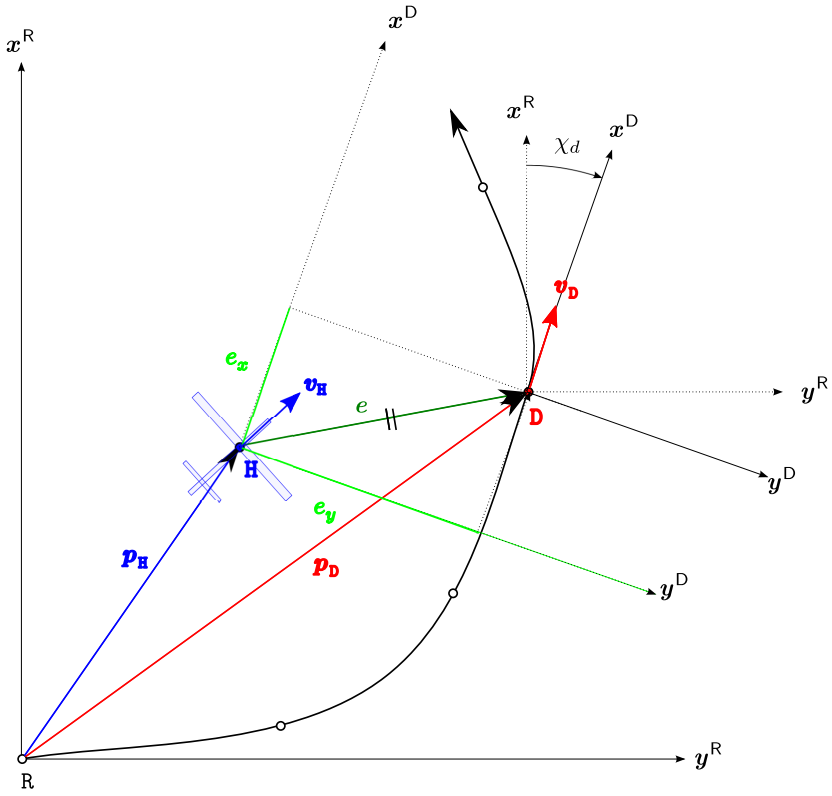



Figure 6.6: Notation used for trajectory tracking, see the first paragraph in §6.2.3 for more details.

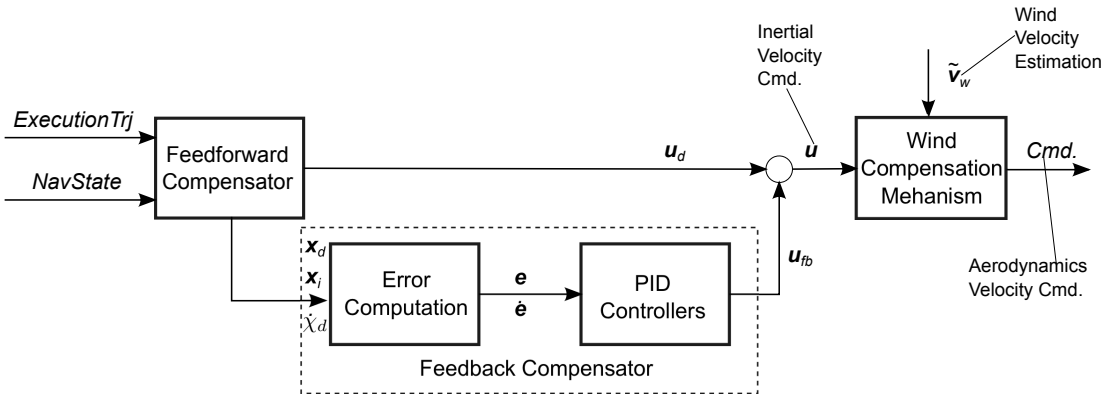



Figure 6.7: 2-Degree of Freedom (DoF) control architecture for the Trajectory Tracker.

vectors resolved in the resolution coordinate system.

Secondly, the feedback compensator corrects for errors between the desired and actual position trajectories. This involves computing the error $\boldsymbol{e} = \begin{bmatrix} e_x & e_y & e_z \end{bmatrix}^T$ between the desired and actual states, and expressing the error in the desired path coordinate system $\mathsf{D}$ (according to the geometry given in Figure 6.6), that is:

$$\begin{aligned} \boldsymbol{e} = \left[\vec{r}_{\mathtt{D}/\mathtt{H}}\right]^{\mathsf{D}} &= \underline{\mathbf{C}}_{\mathsf{R}}^{\mathsf{D}} \left( \underbrace{\left[\vec{\boldsymbol{p}}_{\mathtt{D}/\mathcal{R}}\right]^{\mathsf{R}}}_{\equiv \boldsymbol{p}_{\mathtt{D}}} - \underbrace{\left[\vec{\boldsymbol{p}}_{\mathtt{H}/\mathcal{R}}\right]^{\mathsf{R}}}_{\equiv \boldsymbol{p}_{\mathtt{H}}} \right) \\ &= \underline{\mathbf{C}}_{\mathsf{R}}^{\mathsf{D}} \left( \boldsymbol{p}_{\mathtt{D}} - \boldsymbol{p}_{\mathtt{H}} \right), \end{aligned} \tag{6.2.2}$$

and the error rate $\dot{\boldsymbol{e}} = \begin{bmatrix} \dot{e}_x & \dot{e}_y & \dot{e}_z \end{bmatrix}^T$ can be obtained by taking the derivative of (6.2.2):

$$\dot{\boldsymbol{e}} = \underline{\mathbf{C}}_{\mathsf{R}}^{\mathsf{D}} \left( \dot{\boldsymbol{p}}_{\mathtt{D}} - \dot{\boldsymbol{p}}_{\mathtt{H}} \right) + \dot{\underline{\mathbf{C}}}_{\mathsf{R}}^{\mathsf{D}} \left( \boldsymbol{p}_{\mathtt{D}} - \boldsymbol{p}_{\mathtt{H}} \right), \tag{6.2.3}$$

where, using the *cross-product matrix* and *Poisson's kinematical equations* (see Stevens & Lewis [2003, p.22-28]), the derivative of a rotation matrix can be expressed with:

$$\dot{\underline{\mathbf{C}}}_{\mathsf{R}}^{\mathsf{D}} = -\underline{\boldsymbol{\Omega}}_{\mathcal{D}/\mathcal{R}}^{\mathsf{D}} \underline{\mathbf{C}}_{\mathsf{R}}^{\mathsf{D}} \tag{6.2.4}$$

$$= -\begin{bmatrix} 0 & \dot{\chi}_d & 0 \\ -\dot{\chi}_d & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\chi_d & \sin\chi_d & 0 \\ -\sin\chi_d & \cos\chi_d & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -\dot{\chi}_d \sin\chi_d & \dot{\chi}_d \cos\chi_d & 0 \\ -\dot{\chi}_d \cos\chi_d & -\dot{\chi}_d \sin\chi_d & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{6.2.5}$$

Three proportional-integral-derivative controllers (PID controllers) are then used to correct for the errors:

- *Along-track error* $e_x$ to horizontal speed command $V_{H,fb}$;
- *Off-track error* $e_y$ to ground track command $\chi_{fb}$;
- *Vertical-track error* $e_z$ to **negative**[1] vertical speed command $-V_{V,fb}$.

[1]The positive vertical-track error is pointing downward.

Using the architecture in Figure 6.8 and the parameter values in Table 6.1, the PID feedback controllers significantly reduce the tracking error, as shown in the verification example in Figure 6.9.

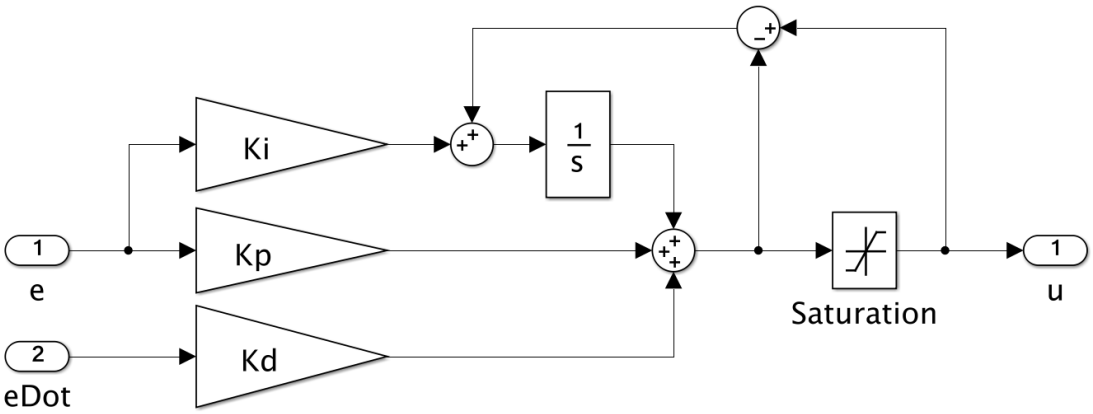


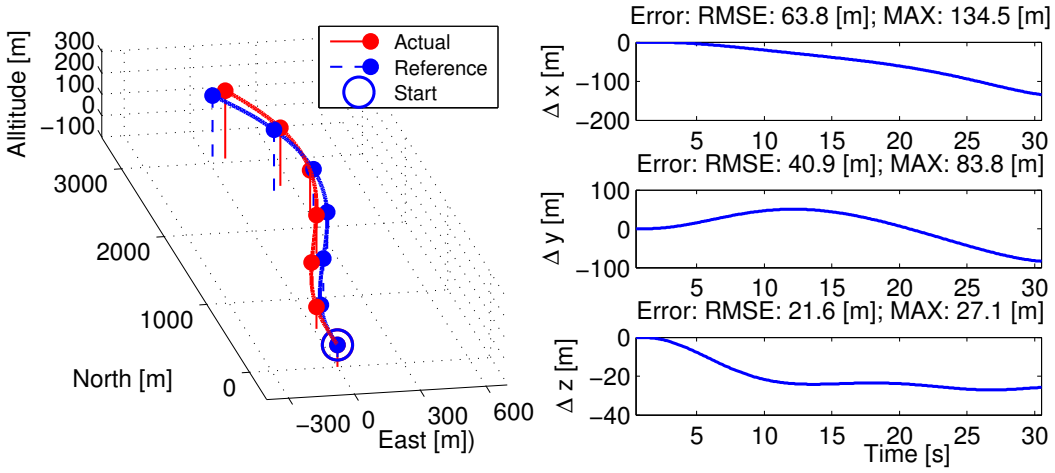

Figure 6.8: Architecture of the PID controller with saturation and anti-windup.

Table 6.1: The parameter values for the PID controllers in the Trajectory Tracker.

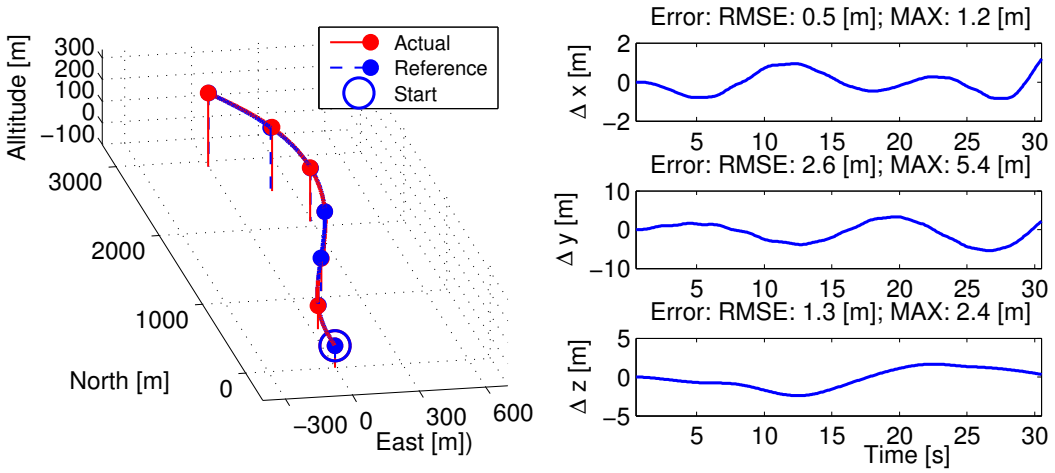
| | $K_p$ | $K_i$ | $K_d$ | *upperlimit* | *lowerlimit* |
|---|---|---|---|---|---|
| $e_x \rightarrow V_H$ | 2 | 1 | 8 | 15 [m/s] | −15 [m/s] |
| $e_y \rightarrow \chi$ | 1 | 0.025 | 5 | 9 [deg] | −9 [deg] |
| $e_z \rightarrow V_V$ | 1 | 1 | 15 | 10 [m/s] | −10 [m/s] |

Finally, it is assumed that a wind compensation mechanism is available to enable the aircraft system to track the inertial velocity command. In this work, the aerodynamic velocity command, as required by the manoeuvre autopilot (see §3.4.3), is obtained by subtracting the estimated wind (assumed to be available) from the inertial velocity command:

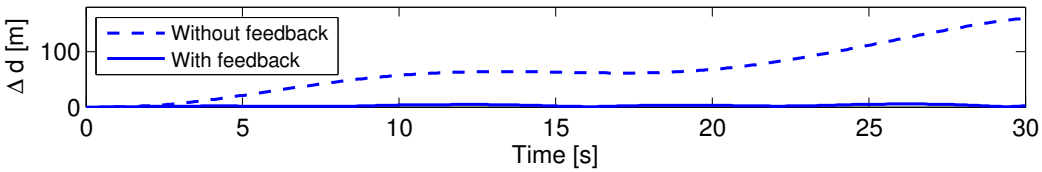
$$Cmd. = \boldsymbol{u} - \tilde{\boldsymbol{v}}_W. \tag{6.2.6}$$




(a) Trajectory tracking without the feedback compensator.




(b) Trajectory tracking with the feedback compensator.




(c) Distance error.

Figure 6.9: Comparision of two resulting trajectories with and without the feedback compensator, validating the capability of the Trajectory Tracker.

## 6.3 Evaluation of the Proposed Approach

### 6.3.1 Safety Performance Evaluation under the Ideal Condition

In order to solely reflect their capability in resolving conflicts, the deliberative and reactive logics were tested with an ideal surveillance condition with no FOR restriction and no estimation error. There are three candidate logics in this evaluation and they were evaluated with 3800 encounters sampled from the standard encounter model. The resulting simulated encounters would have led to an NMAC if the separation between two aircraft was less than 100 ft apart vertically and 500 ft horizontally. The probability of NMAC and the risk ratio were used as the safety performance measures. Table 6.2 summarizes the nomenclature and setting used for this evaluation.

Table 6.2: Experiment settings for the safety performance evaluation.

| Group | Settings/Terms | Value/Description |
|---|---|---|
| **Encounter Samples** | Encounter Geometry | Standard Encounter Set, see Table 4.2. |
| | Sample Size | 3800. |
| | Underlying Risk | Original probability of NMAC is 0.42. |
| **Testing Conditions** | Ideal Condition | See Figure 3.14. |
| | FOR Switch | 0 (No FOR restriction). |
| | Noise Level | 0, see Table 3.7. |
| **Candidate Systems** | Reactive (10Hz) | Geometry-based logic in §3.6. |
| | Reactive (2Hz) | Geometry-based logic in §3.6. |
| | Deliberative (2Hz) | Trajectory-based logic in §6.2. |
| **Performance Measure** | Pr(NM) | Probability of NMAC (4.2.2). |
| | Pr(NM\|E) | Probability of NMAC given an emergency condition (6.3.3). |
| | $RR$ | Risk Ratio (4.2.1). |

**Overall Safety Performance** Table 6.3 summarizes the main results of this evaluation. Without any collision avoidance logic, the probability of an NMAC is $4.22\times10^{-1}$, indicating that about four in ten testing encounters have resulted in an NMAC. All the

Table 6.3: Overall safety performance results of three candidate logics, compared with the original performance with no conflict resolution logic.

| | Deliberative (2Hz) | Reactive (2Hz) | Reactive (10Hz) | Without |
|---|---|---|---|---|
| $NM$ | 27 | 54 | 73 | 1605 |
| Pr(NM) | $7.11 \times 10^{-3}$ | $1.42 \times 10^{-2}$ | $1.92 \times 10^{-2}$ | $4.22 \times 10^{-1}$ |
| $RR$ | 0.017 | 0.034 | 0.045 | 1 |

* Total number of testing encounters is 3800.

probabilities with conflict resolution logics are at least one order of magnitude smaller than the original probability, showing that all candidate logics are effective in preventing NMAC. Comparing the reactive logics' performance, the logic at 2 Hz performed slightly better than that at 10 Hz, which is unexpected, i.e. to achieve a better performance with a lower update rate, and this will be further discussed in the next paragraph. The deliberative logic outperformed the other two reactive logics by at least 50% and further reduced the NMAC probabilities to $7.11 \times 10^{-3}$ and the risk ratio to 0.017.

**Safety Performance in Emergencies** A closer inspection to the resulting NMAC encounters revealed that 1) the intruders in about 95% (i.e. 58 in 61 and 77 in 80) of the NMAC encounters were manoeuvring; 2) in all the NMAC encounters, the linear time-to-CPA $t_{\mathrm{C}_{lin}CD}$, vertical range $r_{V,CD}$ and horizontal range $r_{H,CD}$ (at the time when the conflict is first detected) satisfy either the *time-emergency* condition:

$$t_{\mathrm{C}_{lin}CD} < 15\,\mathrm{s}, \tag{6.3.1}$$

or the *distance-emergency* condition:

$$(r_{H,CD} < 2100\,\mathrm{ft} \wedge r_{V,CD} < 1300\,\mathrm{ft})\,. \tag{6.3.2}$$

However, for the reactive logic at 10Hz, only 59% (43 in 73) of its resulting NMAC satisfy the emergency conditions, indicating that this logic has failed significantly even when the situation is not very "emergent". Without detailed investigation, we cannot conclude the reason for this but it is observed that while the higher rate logic performed better than the lower rate one, in terms of aiming at just missing the intruder with

a 2000 ft safety bubble as designed, it resulted in smaller separation and thus more chance of an NMAC.

To compare the performance in emergency situations, the conditional probability of an NMAC given an emergency condition is calculated with:

$$\Pr(\text{NM}|\text{E}) = \frac{\Pr(\text{NM} \cap \text{E})}{\Pr(\text{E})}. \tag{6.3.3}$$

Tables 6.4 and 6.5 summarize the resulting conditional probabilities of an NMAC given the specified conditions. When the thresholds are very small, both logics were having difficulties to resolve the conflict, about six of ten of these emergency encounters have resulted in an NMAC. As the threshold was increasing, all three logic's safety performance were improving and, normally, the deliberative logic has the largest improvement while the reactive logic at 2Hz has the least. It is worth noting that, when the linear time-to-CPA $t_{C_{lin}CD}$ is less than 6 seconds, the reactive logic at 10 Hz performed better that the reactive logic at 2 Hz and the deliberative logic. It is consistent with the expectation that the higher update can be more effective in resolving conflicts. The performance drop of the deliberative logic can be attributed to its time delay of 0.5 second, when the time delay is comparable to the linear time-to-CPA.

Table 6.4: Safety performance in distance-emergency situations.

| | Pr(NM\|E) with E = $r_{V,CD} < TH_1$ [ft] $\cap$ $r_{H,CD} < TH_2$ [ft] | | | |
|---|---|---|---|---|
| $TH_1$ = | 50 | 200 | 400 | 1300 |
| $TH_2$ = | 750 | 1000 | 2000 | 2100 |
| Reactive (2Hz) | $6.7 \times 10^{-1}$ | $5.0 \times 10^{-1}$ | $2.9 \times 10^{-1}$ | $2.3 \times 10^{-1}$ |
| Reactive (10Hz) | $6.7 \times 10^{-1}$ | $3.5 \times 10^{-1}$ | $2.5 \times 10^{-1}$ | $2.0 \times 10^{-1}$ |
| Deliberative (2Hz) | $6.7 \times 10^{-1}$ | $3.0 \times 10^{-1}$ | $1.7 \times 10^{-1}$ | $9.8 \times 10^{-2}$ |

### 6.3.2 Robustness Analysis on Sensor Noise

In order to evaluate their robustness to feedback error, the three logics were tested with *noisy conditions* subject to four noise levels and no FOR restriction. The same 3800 encounter geometry were used and the risk ratio and probability of unnecessary alert

Table 6.5: Safety performance in time-emergency situations.

| | Pr(NM\|E) with E = $t_{C_{lin}CD} < TH$ [s] | | | |
|---|---|---|---|---|
| $TH$ = | 3 | 6 | 9 | 12 |
| Reactive (2Hz) | $6.4 \times 10^{-1}$ | $5.4 \times 10^{-1}$ | $3.1 \times 10^{-1}$ | $1.7 \times 10^{-1}$ |
| Reactive (10Hz) | $6.4 \times 10^{-1}$ | $3.6 \times 10^{-1}$ | $2.1 \times 10^{-1}$ | $1.3 \times 10^{-1}$ |
| Deliberative (2Hz) | $5.7 \times 10^{-1}$ | $4.4 \times 10^{-1}$ | $1.7 \times 10^{-1}$ | $9.1 \times 10^{-2}$ |

are used as performance measures. Table 6.6 summarizes the nomenclature and setting used for this evaluation.

Table 6.6: Experiment settings for the robustness evaluation.

| **Group** | **Settings/Terms** | **Value/Description** |
|---|---|---|
| **Encounter Samples** | Encounter Set | See Table 4.2. |
| | Total Encounter | 3800. |
| **Testing Conditions** | Noisy Condition | See Figure 3.14. |
| | FOR Switch | 0 (No FOR restriction). |
| | Noise Level | $\{0, 1, 3, 5\}$, see Table 3.7. |
| **Candidate Systems** | Reactive (10Hz) | Geometry-based logic in §3.6. |
| | Reactive (2Hz) | Geometry-based logic in §3.6. |
| | Deliberative (2Hz) | Trajectory-based logic in §6.2. |
| **Performance Measure** | $RR$ | Risk Ratio, see (4.2.1). |
| | Pr(UA) | Probability of Unnecessary Alert, see (4.2.3). |

**Sensitivity Curve** Figure 6.10 summarizes the main evaluation results in the sensitivity curves of the three logics, illustrating the effect of varying the noise level on the Risk Ratio and on the probability of unnecessary alert rate. Each point on the sensitivity curves was estimated using 4 independent sets of 950 simulated encounters (i.e. randomly divide 3800 simulated encounters into 4 sets). The error bars indicate the standard deviation of the estimates over the four sets.

It can be seen that:

1. using the same conflict detection logic, the two Pr(UA) sensitivity curves of lower update rate are exactly the same and the curve of higher rate is with larger unnecessary alert probabilities as there is a greater chance to trigger an alert by the noisy estimation;

2. the risk ratio is relatively insensitive to the noise until a level of about four and there is no appreciable difference between the nominal case (noise level 1) and the case without any uncertainty. This trend is consistent with the results by Edwards [2012] in a parametric analysis of radar position uncertainty. Moreover, for the reactive logic at 10Hz, the results with little noise have better performance than those with no noise at the cost of more unnecessary alerts; and

3. the deliberative logic always outperformed the other two reactive logics, no matter whether in terms of the safety performance itself or the performance robustness to the noise level. In particular, as the noise level (equivalent to the feedback error) was increasing, the performance and its robustness of the deliberative logic performed overwhelmingly better than the reactive ones'.

### 6.3.3 Evaluation of the Field-of-Regard-Restriction Effect

In order to evaluate the FOR-restriction effect on safety performance, the three logics were tested with the *noisy conditions* (with noise but no FOR restriction) and the *realistic conditions* (with noise and FOR restriction), subject to four different noise levels, using the same 3800 encounter geometry. Different measures to evaluate the safety performance, FOR-restriction effect, operational suitability and feedback quality are used and will be described as the results are presented. Table 6.7 summarizes the nomenclature and setting used for this evaluation.

**FOR Effect on Safety Performance** Figure 6.11 summarizes the overall number of the resulting NMACs (denoted by $NM$) for each of three logics in the cooperative and non-cooperative cases. The bar chart also decomposes the overall $NM$ into three components: the number of Unresolved NMAC ($UN$), Induced NMAC ($IN$) and Miss Detection ($MD$). Compared with their cooperative counterparts, the non-cooperative cases not only have more NMACs due to miss detections but also have

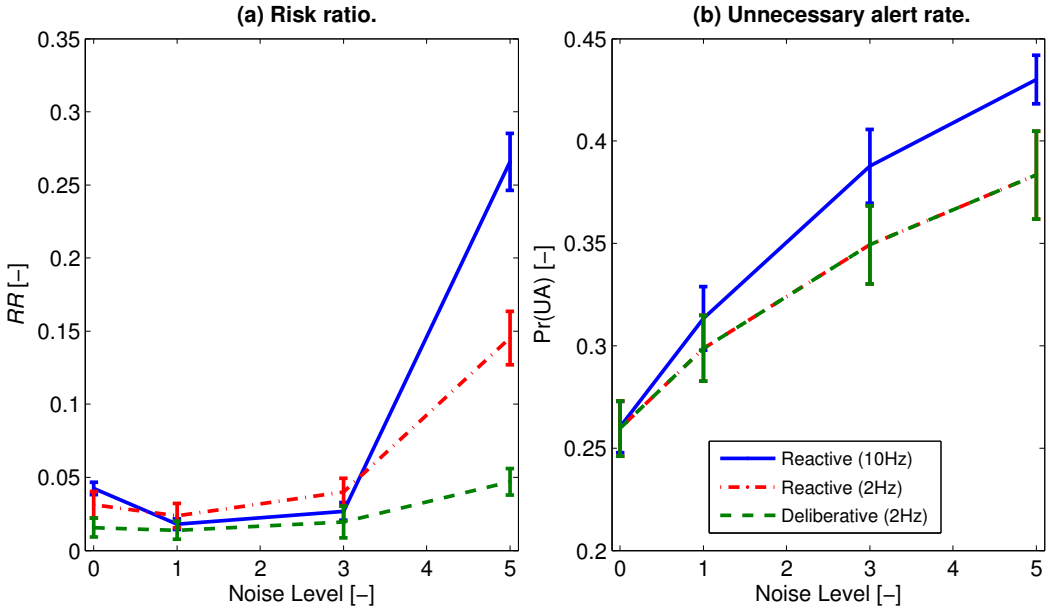



Figure 6.10: Deliberative and reactive logics' sensitivity curves on Risk Ratio while varying the feedback error level.

significantly more unresolved NMACs, mainly due to late detections and erroneous estimation caused by the FOR restriction.

Furthermore, purely to compare the performance of the conflict resolution logics, the miss detection encounters are excluded and the number of the *resolution-related* NMACs is calculated:

$$NM_{res} = UN + IN = N - MD. \tag{6.3.4}$$

Figure 6.12 shows the sensitivity curves of $NM_{res}$, for each of three logics in the cooperative and non-cooperative cases. It can be seen that the shapes of two curves are similar, indicating there is little impact of FOR on their robustness to noise; and the deliberative logic also achieved the best safety performance and robustness to noise level in the realistic conditions.

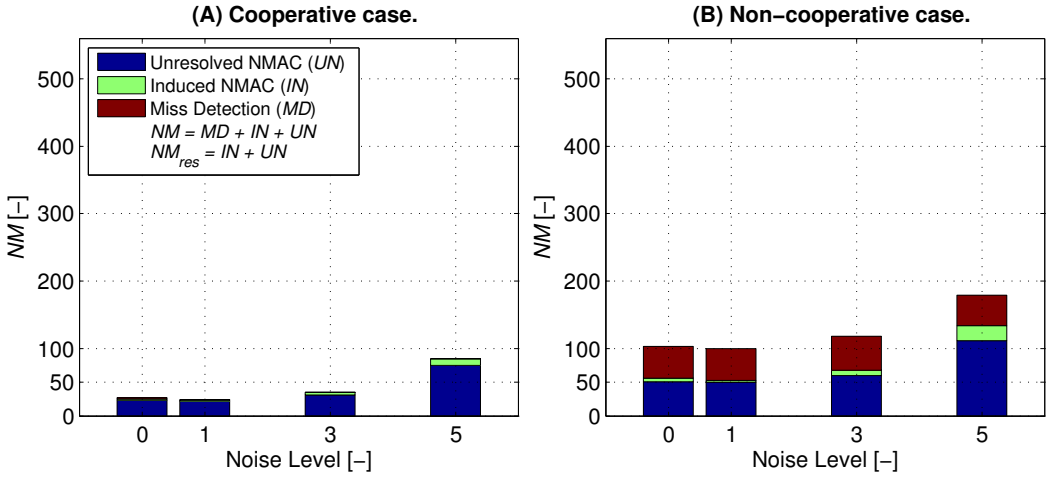
Lastly, to assess the impact of the FOR restriction on each candidate's safety performance, the *FOR-induced* NMAC event is defined, which is an encounter that leads to an NMAC in the non-cooperative case but does not in the cooperative case. The

Table 6.7: Experiment settings for the evaluation of FOR effect.

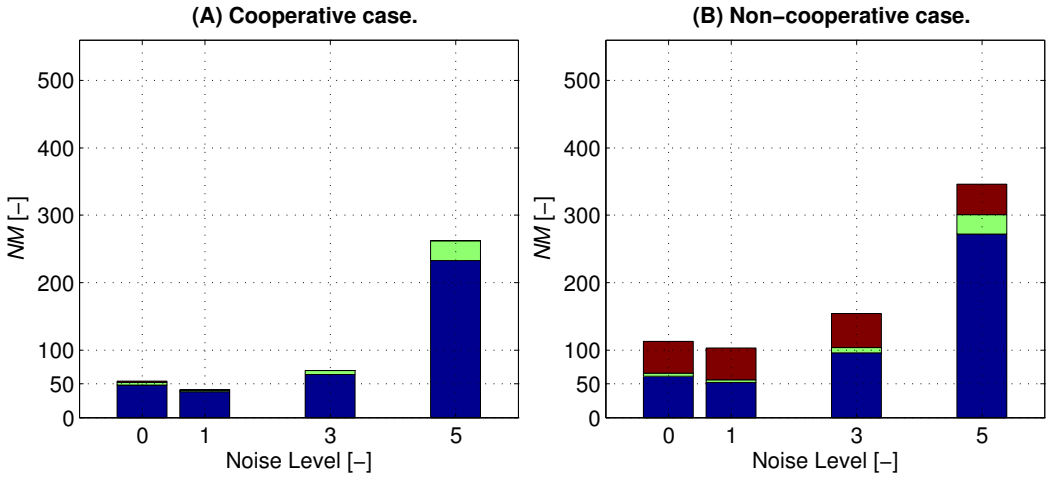
| Group | Settings/Terms | Value/Description |
|---|---|---|
| **Encounter Samples** | Encounter Set | See Table 4.2. |
| | Total Encounter | 3800. |
| **Testing Conditions** | Realistic and Noisy Condition | See Figure 3.14. |
| | FOR Switch | 0 and 1. |
| | Noise Level | $\{0, 1, 3, 5\}$, see Table 3.7. |
| **Candidate Systems** | Reactive (10Hz) | Geometry-based logic in §3.6. |
| | Reactive (2Hz) | Geometry-based logic in §3.6. |
| | Deliberative (2Hz) | Trajectory-based logic in §6.2. |
| **Performance Measure** | $NM$ | Number (Num.) of NMAC. |
| | $NM_{res}$ | Num. of resolution-related NMAC (6.3.4). |
| | $NM_{FOR}$ | Num. of FOR-induced NMAC, see page 134. |
| | E $(\lambda_{ava})$ | Expected feedback availability (4.3.7). |
| | E $(\lambda_{err})$ | Expected feedback error (4.3.8). |
| | Pr(CoC) | Probability of clear of conflict (4.3.6). |
| | Pr(SC) | Probability of secondary conflict (4.3.5). |

number of FOR-induced NMACs, denoted by $NM_{FOR}$, is calculated and the higher the $NM_{FOR}$, the more impact it has on the safety performance. Figure 6.13 shows the $NM_{FOR}$ sensitivity curves of the three logics. It can be seen that the curve of the deliberative logic has the most $NM_{FOR}$ in all noise levels, indicating that the deliberative logic is affected by FOR restriction the most. This observation is unexpected as the deliberative logic, embedding a planning process, is expected to be less reliant on the continuous feedback and thus to be less sensitive to the FOR restriction. This will be further discussed in the next section.

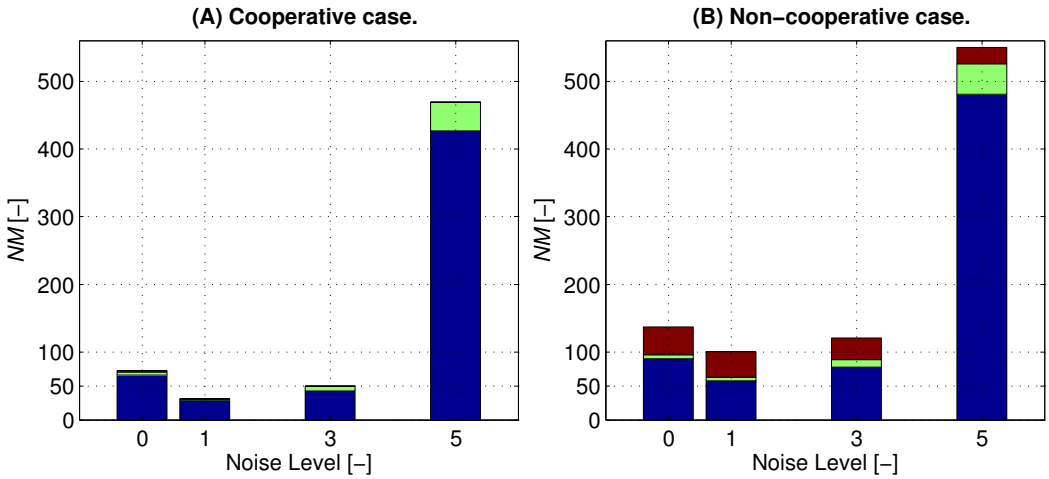
**FOR Effect on Feedback Quality** Two metrics were introduced in §4.3 to measure the feedback quality: the expected feedback availability E $(\lambda_{ava})$ and the expected feedback error level E $(\lambda_{err})$. Figures 6.14 and 6.15 summarise and compare the sen-




(a) Deliberative logic at 2 Hz.




(b) Reactive logic at 2 Hz..




(c) Reactive logic at 10 Hz..

Figure 6.11: NMAC events decomposition for the three candidate logics.

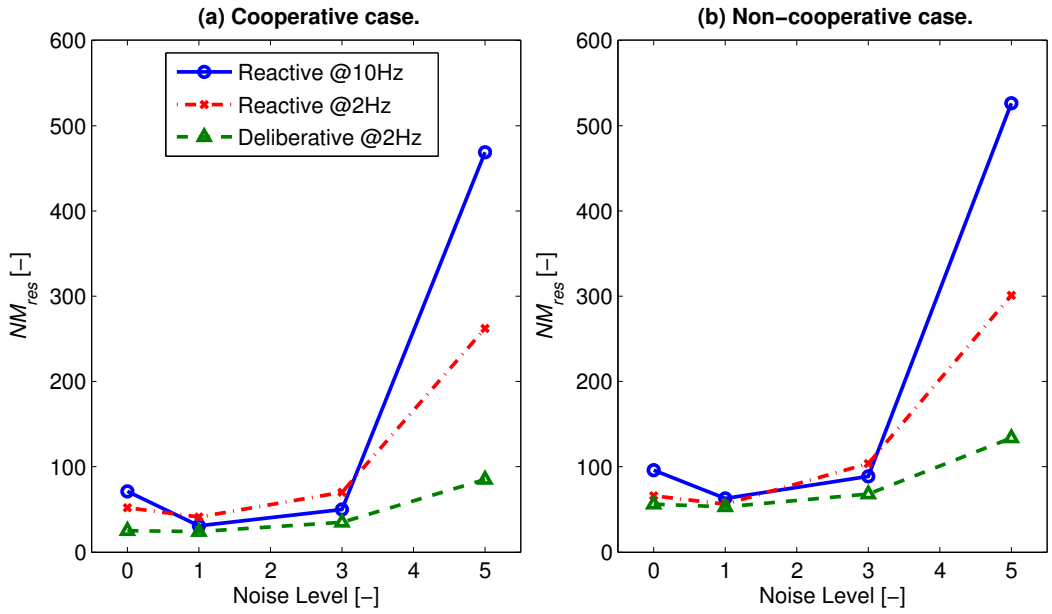



Figure 6.12: Sensitivity curves of probability of resolution-related NMAC, while varying the noise level.

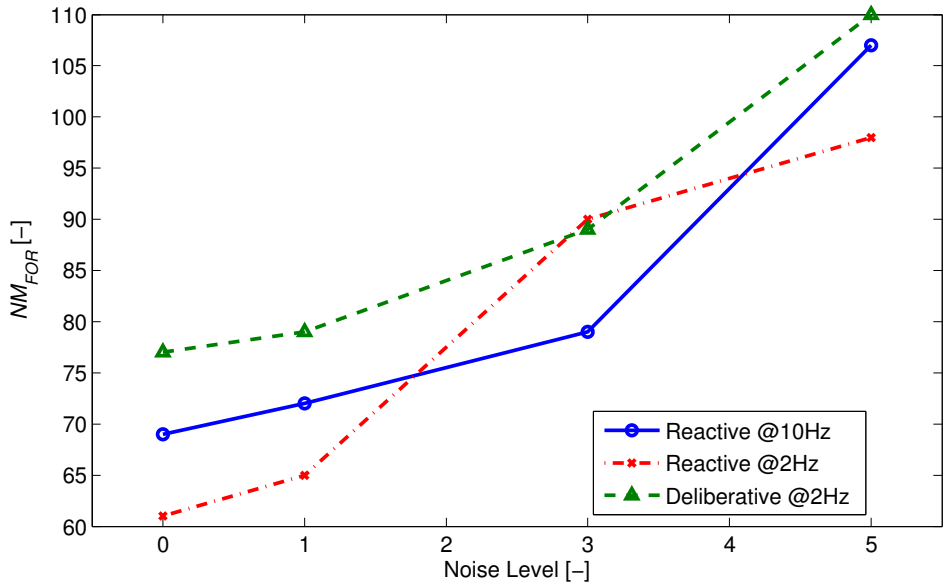



Figure 6.13: Sensitivity curves of probability of FOR-related NMAC, while varying the noise level.

sitivity curves of the E ($\lambda_{ava}$) and E ($\lambda_{err}$), for the three logics in both cooperative and non-cooperative cases.

In the cooperative case, the feedback is always available and therefore the feedback error is proportional to the noise level. This trend remains the same in the non-cooperative case. Moreover, in the non-cooperative case, the feedback availability were decreased from 1 to about 0.9 and to about 0.75 for the reactive and deliberative logic, respectively, due to the FOR restriction. This decrease in availability has led to the increase in the feedback error, which is manifested in the deliberative logic's feedback error level.

**FOR Effect on Operational Suitability** Two metrics were introduced in §4.3 to measure the operational suitability: the probability of clear of conflict Pr(CoC) and the probability of secondary conflict Pr(SC). The bigger the Pr(CoC) (or the smaller the Pr(SC)), the better the system performance is.

Figures 6.16 and 6.17 summarise and compare the sensitivity curves of the Pr(CoC) and Pr(SC), for the three logics in both cooperative and non-cooperative cases. The Pr(CoC) figure shows the typical trend that the performance would decline in the non-cooperative case and this decline becomes more severe as the noise level increases; while the Pr(SC) figure shows the opposite that the better performance is found in the non-cooperative case, i.e. with a smaller probability of secondary conflict. This is because there are more chances to trigger the conflict detection logic by the noisy feedback in the cooperative case.

More importantly, the figure shows that the deliberative logic has a higher probability of Clear of Conflict and smaller probability of Secondary Conflict than the baseline method, indicating that the deliberative logic can choose a safer terminal state to issue a CoC.

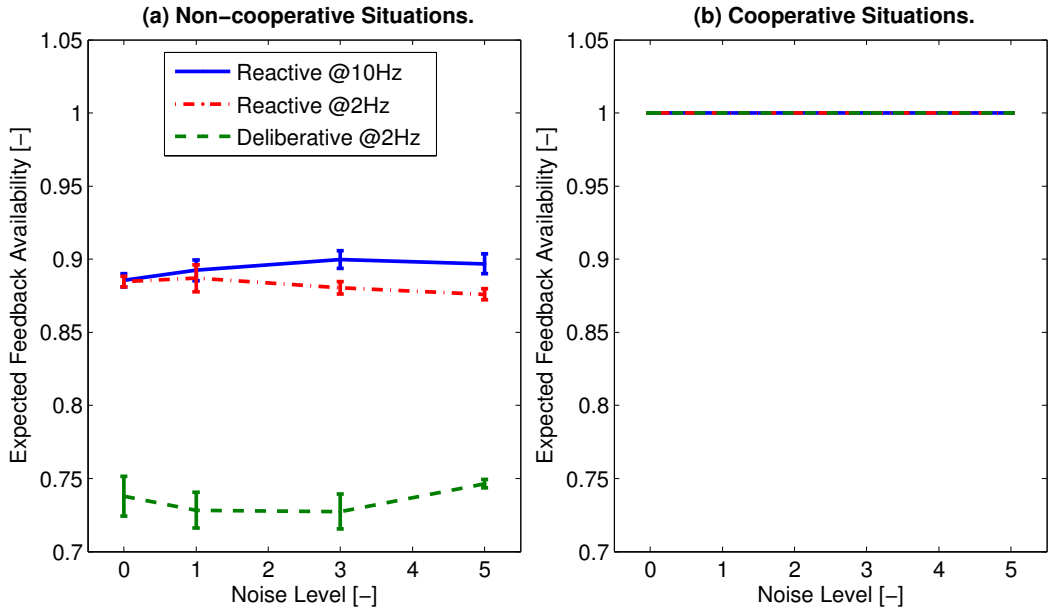



Figure 6.14: Sensitivity curves of expected feedback availability level, while varying the noise level.

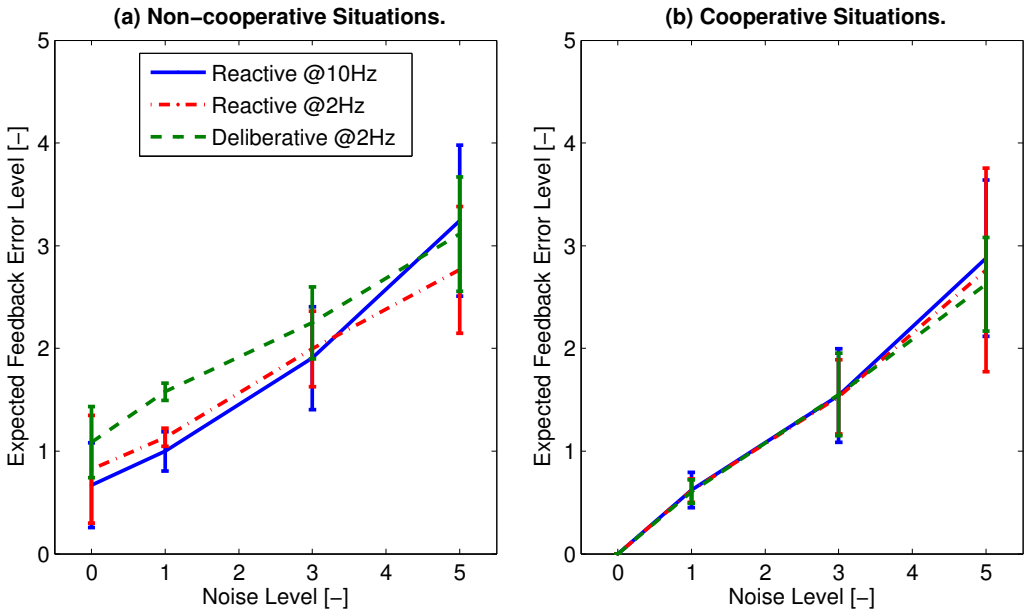



Figure 6.15: Sensitivity curves of expected feedback error level, while varying the noise level.

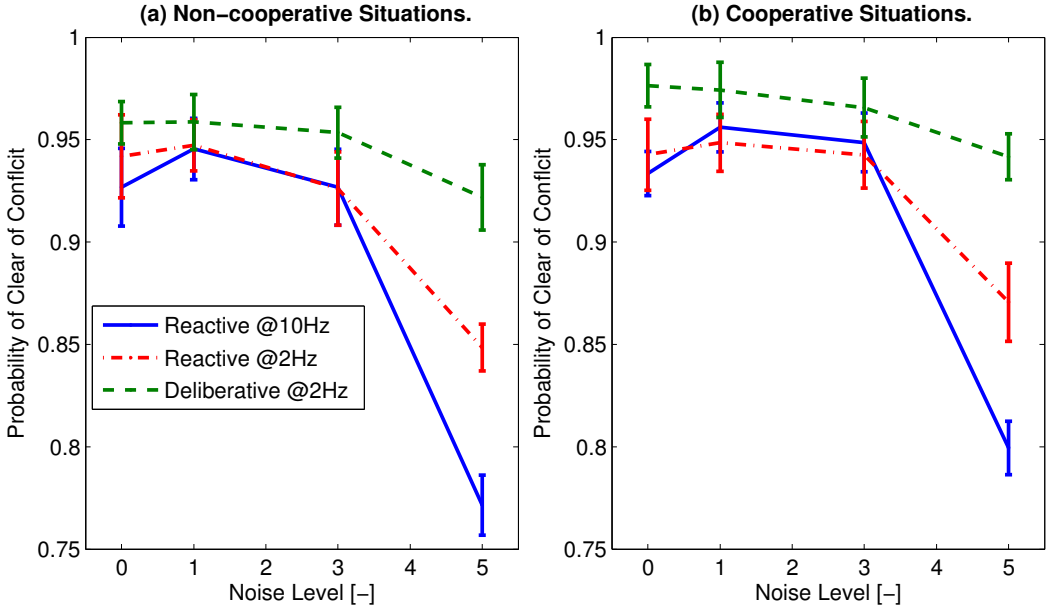



Figure 6.16: Sensitivity curves of probability of clear of conflict, while varying the noise level.

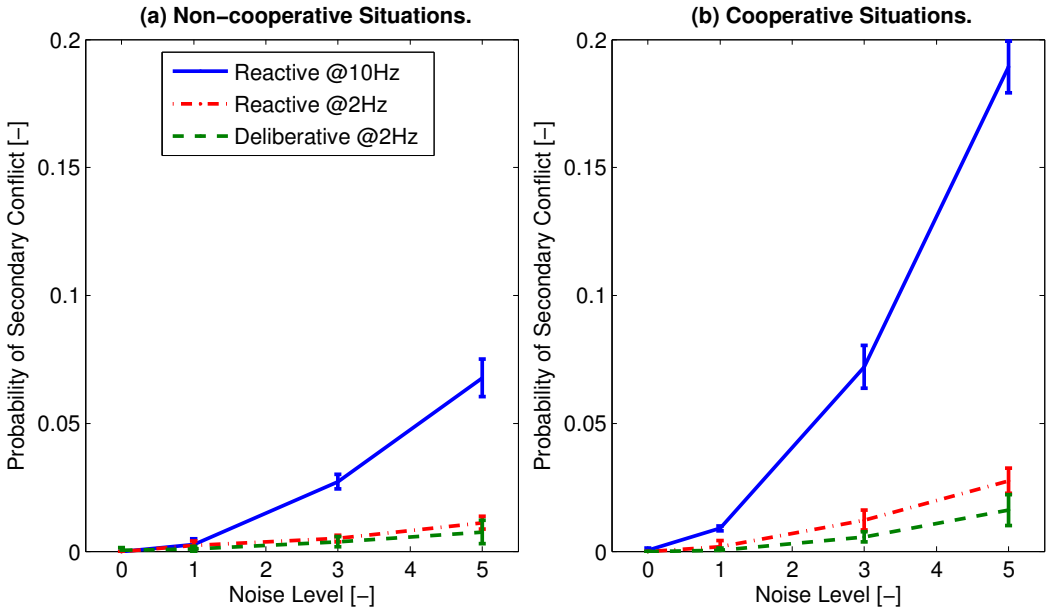


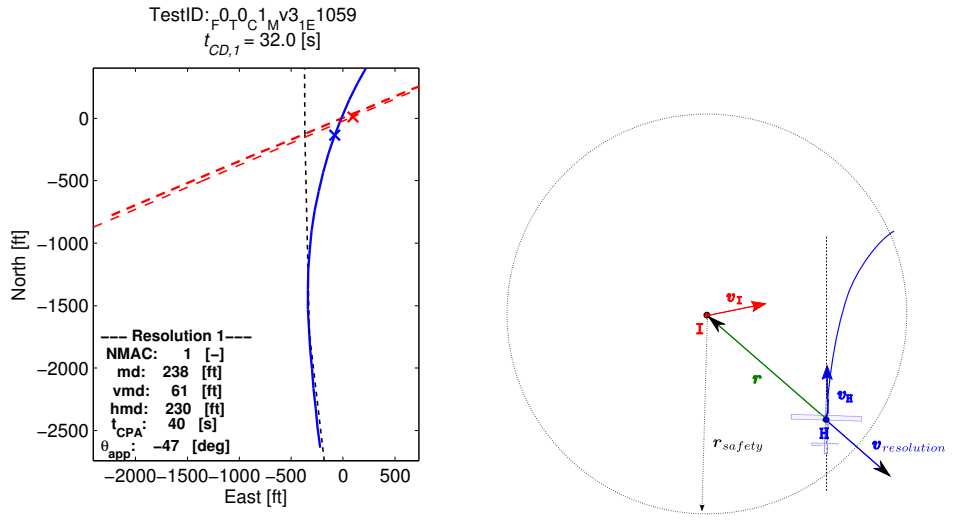

Figure 6.17: Sensitivity curves of probability of secondary conflict, while varying the noise level.

## 6.4 Discussions with Example Encounters

This section performs a closer inspection of four phenomena—the ineffectiveness of the emergency strategy, tendency to touch the protected volume, parallel-track encounters and field-of-regard effect—observed from the analysis of the resulting NMACs.

### 6.4.1 Ineffectiveness of the Emergency Strategy

As described in the conflict resolution logic on page 65, if the relative range of the intruder is smaller than the radius of the safety bubble (2000 ft in the current implementation), an emergency strategy will be used. Figure 6.19 shows an example encounter in which the emergency strategy is used but failed. As can be seen from the result of the reactive logic in figure (a), despite the resolution command to turn right at about 32 seconds, the close encounter still resulted in an NMAC at 40 seconds. Figure 6.18 shows the detail of the encounter geometry and explanation of the decision to turn right in the emergency strategy. On the other hand, the deliberative logic in Figure 6.19b succeeds in preventing the eight-second-ahead NMAC by issuing a relatively complex resolution to simultaneously turn left, climb and decelerate.




(a) Closer look of the resulting NMAC geometry.

(b) Resolution decision of the emergency strategy.

Figure 6.18: Details of the example encounter in Figure 6.19, while using the reactive logic's emergency strategy.

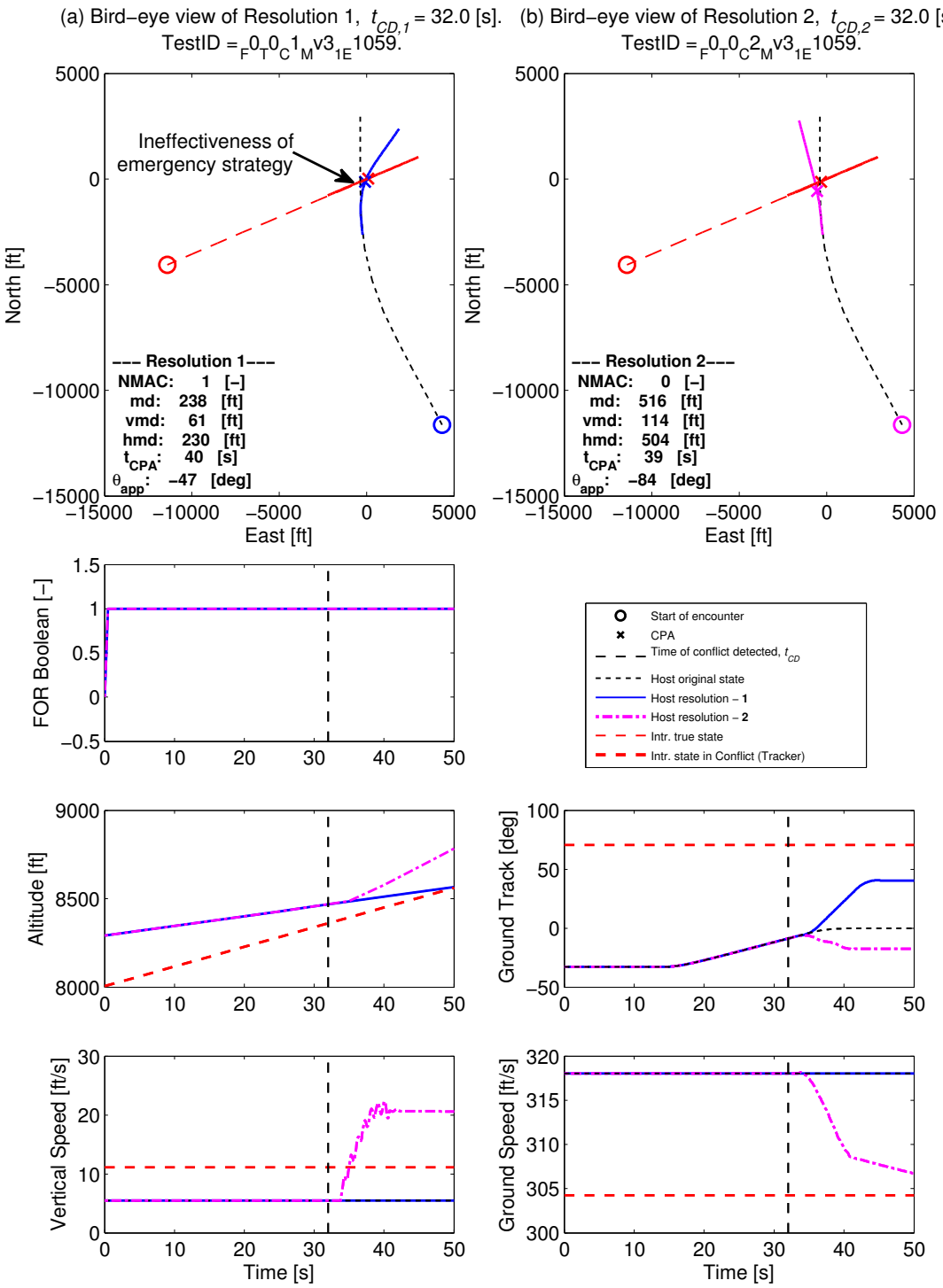
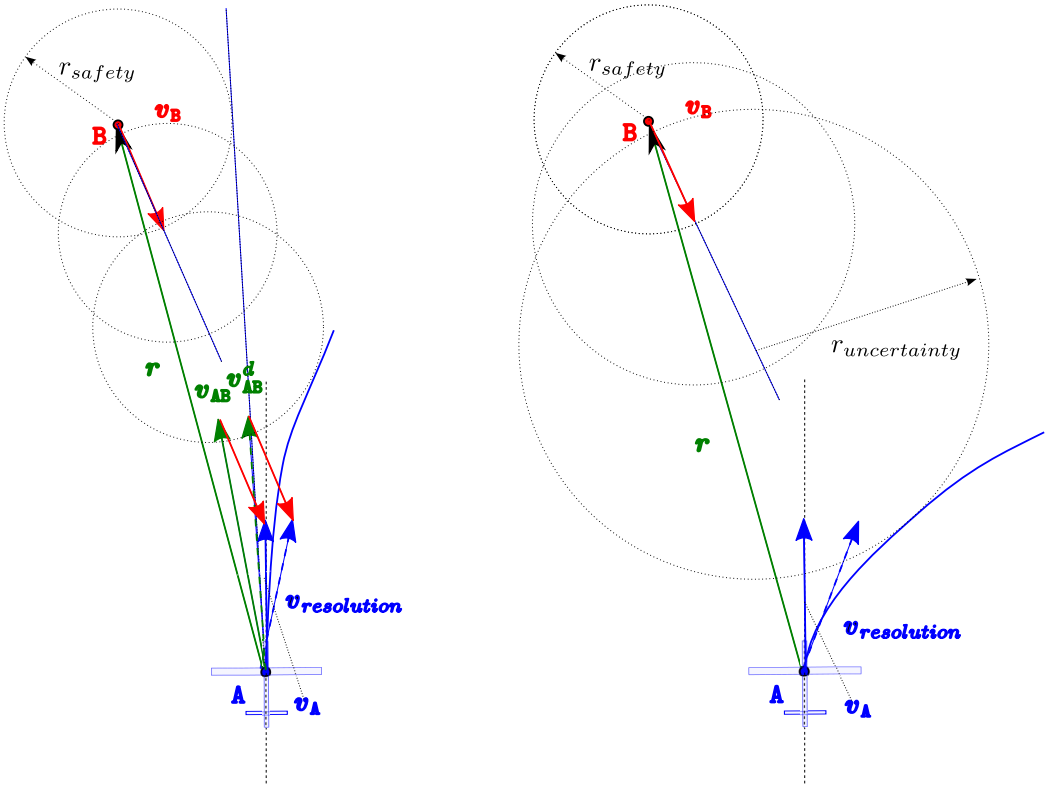



Figure 6.19: Example encounter illustrating a failure case when using the reactive logic's emergency strategy. Resolution 1 is the reactive logic in cooperative cases. Resolution 2 is the deliberative logic in cooperative cases.

### 6.4.2 Conservativeness in Uncertainty Propagation

The deliberative logic uses a more conservative approach to propagate the intruder uncertainty than the reactive logic. The deliberative logic uses an uncertainty bound that is growing with time and the resulting resolution manoeuvre tends to move away from the area to which the intruder is possibly travelling, as shown in Figure 6.20a. Figure 6.21 shows an example encounter and compares the resulting resolution trajectories of the two logics. Compared with the reactive logic's resolution trajectory (from the bird-eye view and the vertical profile), the deliberative logic has resulted in not only larger separation with the intruder but also more deviation from the original route. Although an optimization process in the deliberative logic is applied to minimize the route deviation (and other cost), the near-optimal solution obtained from the trajectory planning algorithm would be inevitably subject to this conservativeness.

Despite the possibility of larger route deviation, it pays to be conservative in terms of robustness. With this conservative approach to propagate uncertainty, the deliberative logic is shown to be more robust to high level of intruder uncertainty (see Figure 6.10). While this empirical approach is chosen because of its simple implementation (i.e. it uses a fixed set of empirical parameters to control the uncertainty margin and its growing rate), there exist other approaches to propagate the uncertainty in a more systematic manner (such as the probabilistic approach Kochenderfer *et al.* [2011] and the worse-case approach based on the level set Bayen *et al.* [2003]). Utilizing the more systematic approach should be able to provide more accurate uncertainty propagation at the cost of additional computation. The possible performance gain by utilizing the more systematic approach is left for future study.




(a) Reactive logic.

(b) Deliberative logic.

Figure 6.20: Comparison of two uncertainty propagation approaches.

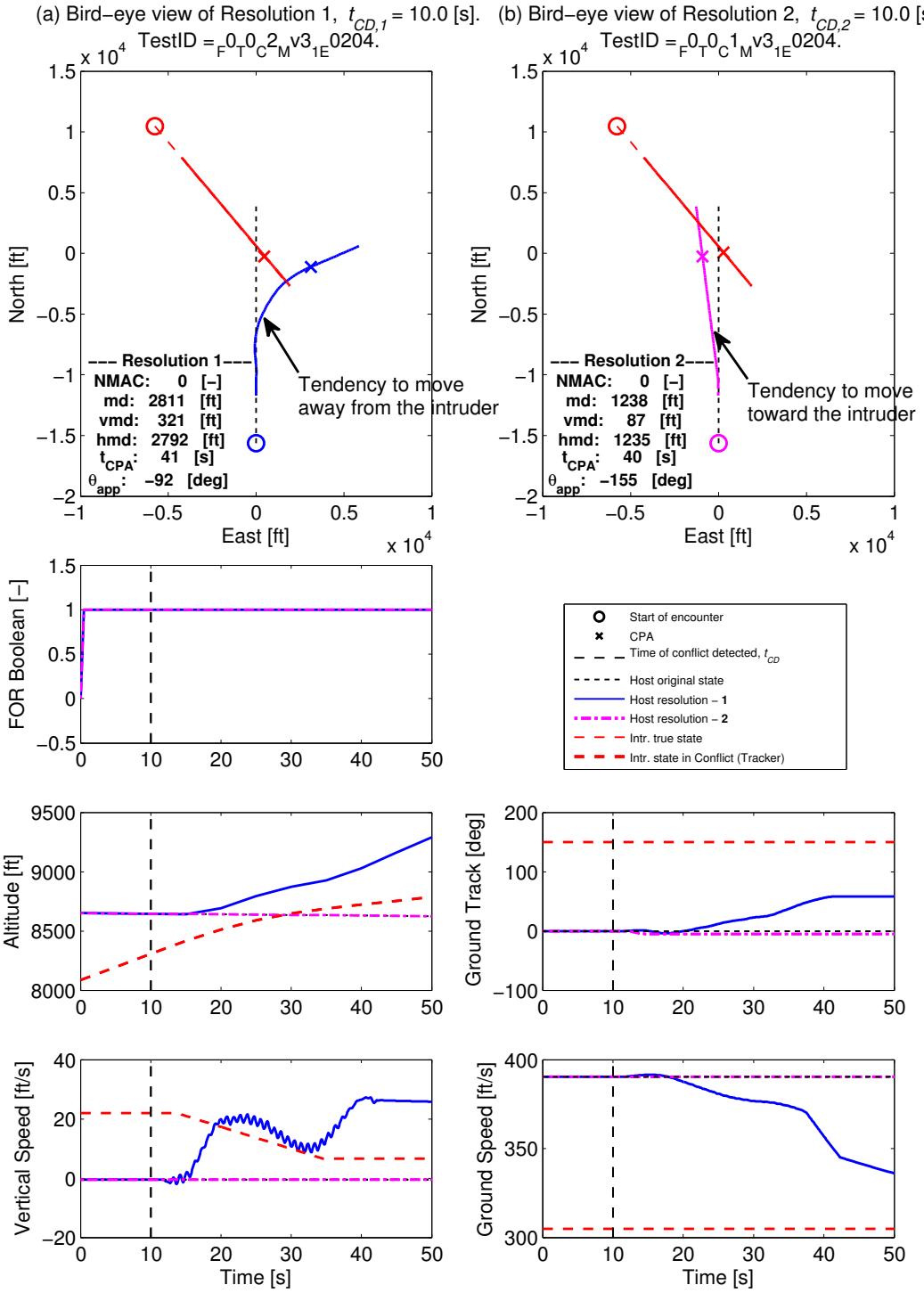



Figure 6.21: Example encounter illustrating the conservativeness of the uncertainty propagation approach used. Resolution 1 is the deliberative logic and Resolution 2 is the reactive.

### 6.4.3 Parallel-Track Results

Among the failures cases of the reactive logic, there is a type of resulting NMAC encounter, referred to as *parallel-track* results, that accounts for a considerable proportion of the resulting NMACs in our testing. Figure 6.23 shows one example encounter of the parallel-track results. As can be seen from the Resolution 1's bird-eye view and ground track profile, the reactive logic, considering an intruder coming from behind on the left-hand side, commanded a resolution manoeuvre to turn right; however, given that particular geometry, the host ended up travelling in parallel to the intruder at about 30 seconds. In this parallel situation, the geometric solution becomes ineffective and finally resulted in an NMAC at about 42 seconds.

To summarize, Figure 6.22 depicts the observed casual chain from the particular "near-parallel" geometry to the resulting NMAC encounters, for both the cooperative and non-cooperative cases. It is worth noting that, according to the rules of the air, the overtaking traffic, which is supposed to have the other aircraft in his FOR, should give way to avoid the NMAC in those situations.

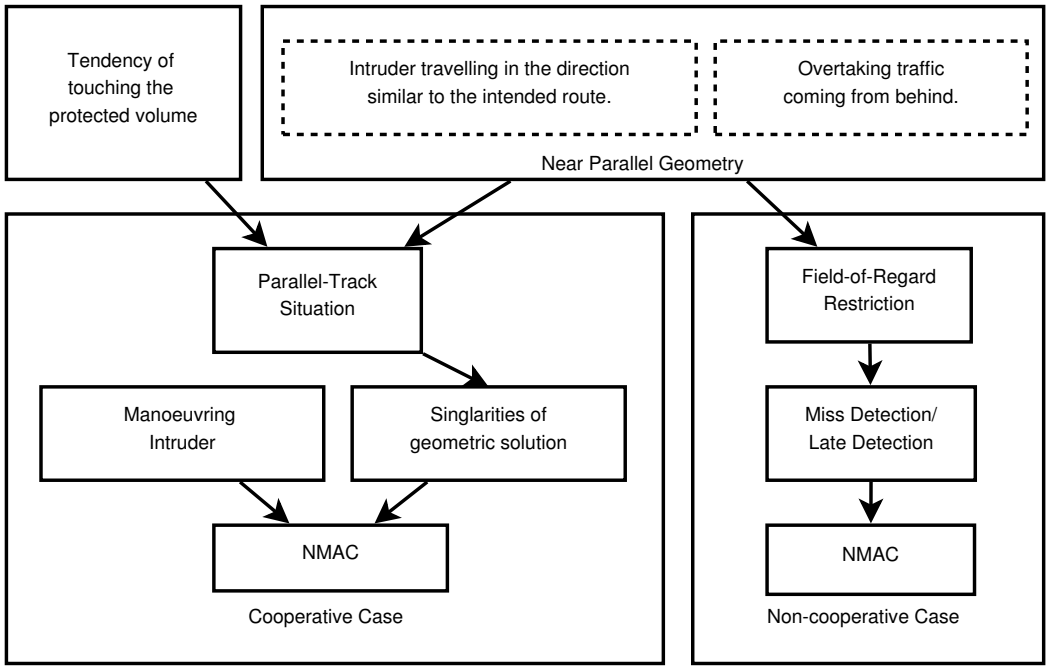



Figure 6.22: An observed causal chain from the near-parallel geometry to the resulting NMAC encounters.

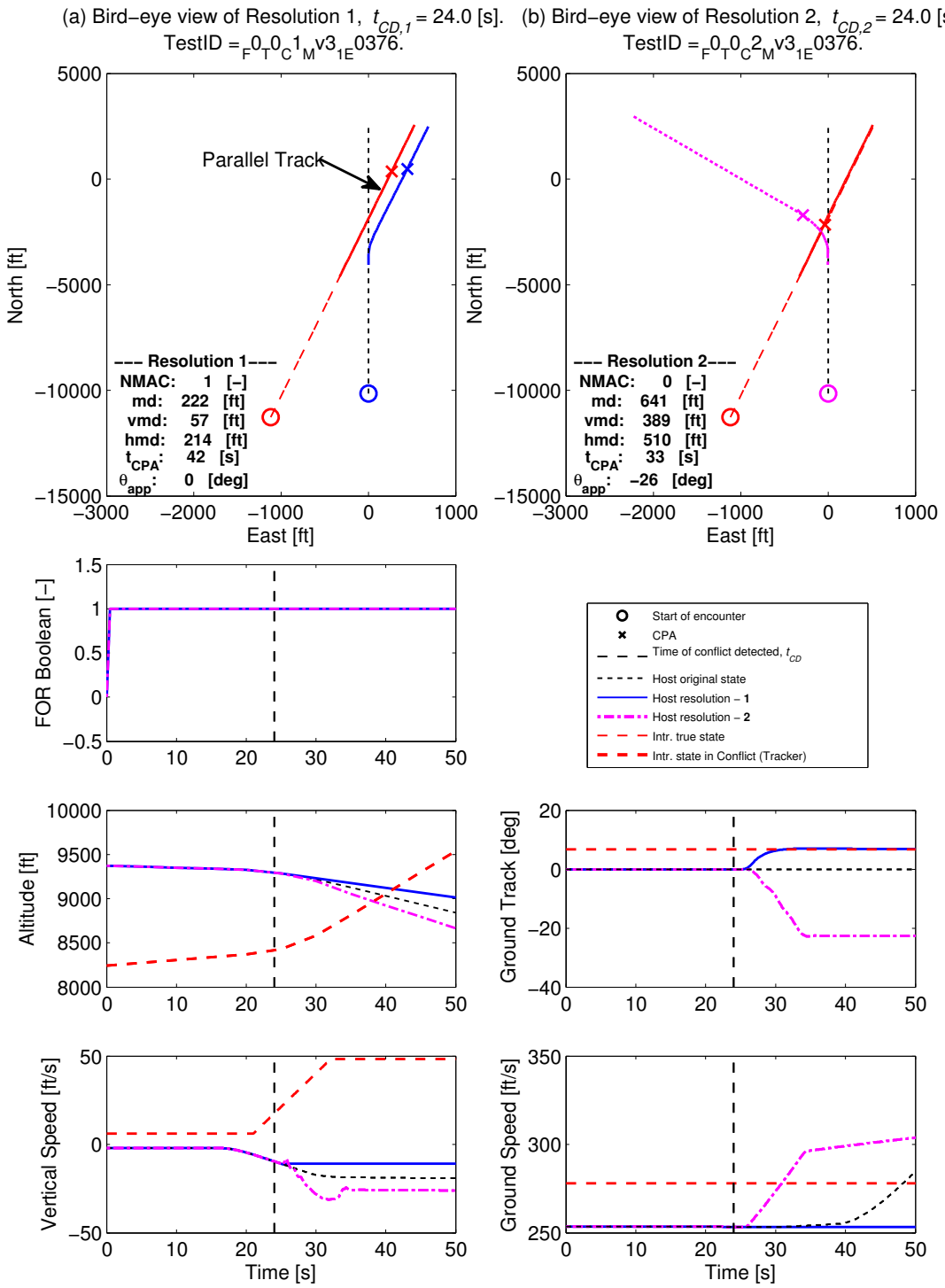



Figure 6.23: Example encounter illustrating the **parallel-track** result. Resolution 1 is the reactive logic in non-cooperative cases. Resolution 2 is the deliberative logic in non-cooperative cases.

### 6.4.4 Field-of-Regard-Restriction Effects

This section discusses two initially unexpected observations found in §6.3.3: 1) despite the introduction of the *intruder invisibility cost* in the trajectory planner (see 5.2.4), it is found in Figure 6.15 that the feedback availability with the deliberative logic is worse than those with the reactive logics; and 2) although the overall performance of the deliberative logic, under the FOR restriction, is better than those of the reactive logics, it appears in Figure 6.13 that the FOR restriction has more impact on the deliberative logic's safety performance than on the reactive logics'.

Figure 6.25 shows an example encounter illustrating the effect of FOR restriction. Compared with those of the Resolution 1 (reactive logic), the Resolution 2's vertical speed, ground track and ground speed profiles show that the resolution from the deliberative logic required more manoeuvring, so that more separation (as shown in the bird-eye view and the altitude profile) could be achieved to accommodate the uncertainty caused by the high level of noise (as shown in the bird-eye views). As a result of more manoeuvres, the intruder was beyond the host FOR more often; and as annotated in the Figure 6.25, the error in the estimation of the intruder's position also became significantly larger as the intruder was beyond the FOR. The above observations are explained by the observed causal chain as shown in Figure 6.24.

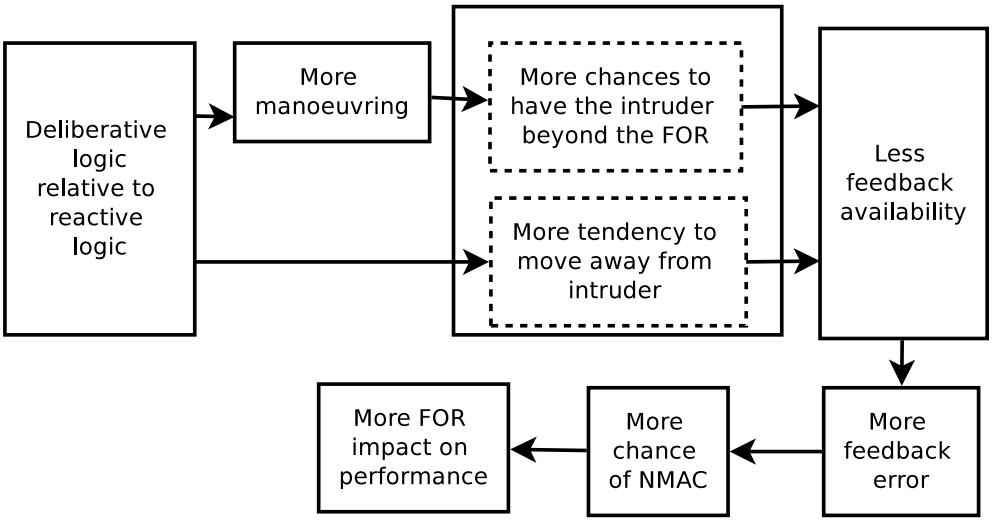



Figure 6.24: An observed causal chain illustrating the effect of FOR restriction on the feedback quality and safety performance.

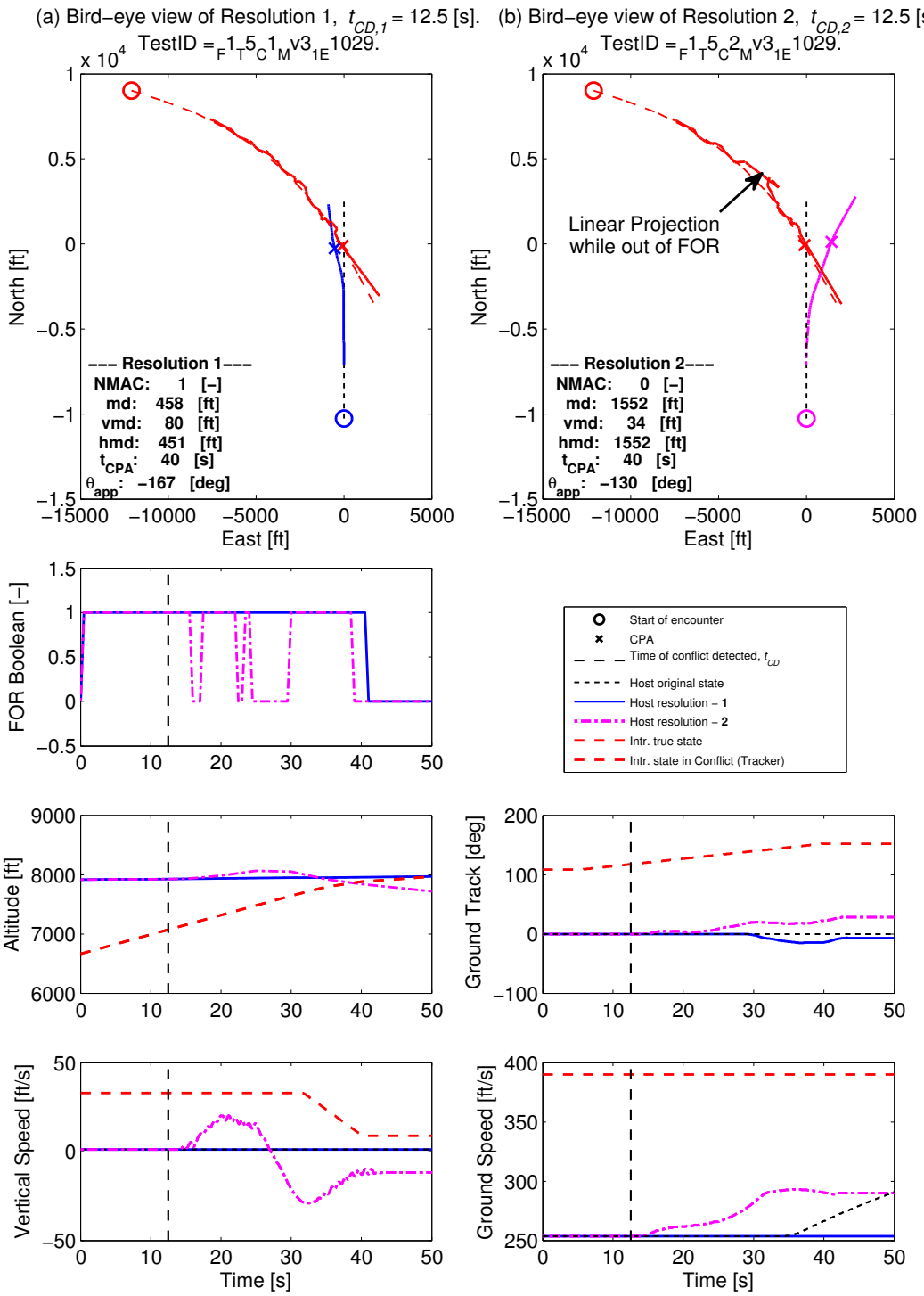



Figure 6.25: Example encounter illustrating the **FOR effect**: 1) the effect of manoeuvring on feedback availability; and 2) the effect of feedback availability on feedback error. Resolution 1 is the reactive logic (at 2Hz) and Resolution 2 is the deliberative logic. Both are in non-cooperative cases with noise at level 5.

## 6.5 Summary

This chapter has presented the three-layer architecture used to integrate the trajectory planner into the collision avoidance logic. The three components in each layer have been described in detail with their interfaces and concepts of operations. The examples for each component to verify their implementations have also been shown. In particular, the trajectory planner has been implemented in MATLAB executables and the computation time has been reduced by 10 times to about 0.5 seconds.

The integrated prototype has been tested with Monte Carlo simulations with 3800 encounter geometries under 8 different surveillance conditions (resulting in 30400 test cases in total) and the results have been compared with those of the baseline logic, which uses the geometric method in a reactive manner as opposed to the deliberative manner of the integrated system. It is encouraging to observe that the deliberative conflict resolution logic, despite a 0.5 second latency, has better safety performance under all the testing surveillance conditions and it is significantly more robust to the sensor noise.

Furthermore, closer inspection of the resulting NMAC encounters indicates that 1) most failure cases of the baseline reactive logic is attributed to the ineffectiveness of its emergency resolution strategy; 2) the deliberative logic with a conservative approach to propagate the uncertainty achieved better robustness at the cost of larger route deviation; 3) the deliberative logic, using a growing uncertainty bound around the intruder, significantly outperformed the reactive logic in the near-parallel-track situations; and 4) the deliberative logic achieved better safety and robustness performance even when its feedback quality is worse than those of the reactive logic (due to larger degree of manoeuvring), indicating a larger degree of robustness to feedback error.

It is worth noting that the sample size of testing encounters is small compared with that required the international standard [ICAO, 2007] and they are obtained under the following conditions: 1) no wind, 2) no navigation error, 3) assuming a global coordinate is available, and 4) with ideal autopilot dynamic of the first order. As the performance of the trajectory-based logic is expected to rely on the accuracy of the navigation state and the trajectory tracking performance considerably, the robustness to these uncertainties would need to be further assessed.

# Chapter 7

# Conclusions

The aim of this thesis is to develop a real-time trajectory planning algorithm for a novel UAV collision avoidance logic and to determine the integrated logic's feasibility, merits and limitations for practical applications. This chapter draws conclusion on the proposed logic's feasibility, merits and limitations, outlines the main deliverables of this work, highlights the contributions to knowledge and suggests areas for further development.

## 7.1 Feasibiliy, Mertis and Limitations

Table 7.1 summarizes the objectives, findings and conclusions of the five simulation experiments presented in this thesis. These are the underpinnings to support the feasibility and potential benefits of the proposed approach. The areas that required further research will also be highlighted in this section.

**Feasibility** The main result of the initial feasibility study is summarized in Table 7.2, from which it can be seen that:

- The proposed trajectory planning method is able to achieve an update rate of 2Hz with a standalone executable running in Simulink in the laptop environment. For reference, ACAS X is expected to operate at 1Hz [Kochenderfer *et al.*, 2012] and MIDCAS [2012] targets at a update rate of 5Hz for collision avoidance.

- The testing results shows that the proposed approach, despite a 0.5 seconds computation latency for trajectory planning, outperformed the baseline approach in terms of safety performance and robustness to sensor and feedback error. The baseline approach is drawn from the literature and its effectiveness is demonstrated with a flight test campaign presented in Luongo *et al.* [2011].[1]

- The resulting risk ratios of the proposed approach is comparable with the reference threshold used in the example feasibility analysis given by Cole *et al.* [2013].

**Merits** Based on the data given in Table 7.1 and the author's opinions, the following potential benefits of the proposed approach are expected:

- With the capability to plan and perform a larger range of high performance avoidance manoeuvres (i.e more resolution actions available), the proposed approach should have better safety performance, especially when the time or distance separation is small.

- Given more resolution actions, the proposed approach could issue a more timely collision alert than the methods with less resolution alternatives could. This could reduce the nuisance alerts by filtering the early false alarms caused by noisy measurement and/or uncertain intruder motion.

- The proposed approach could achieve better robustness to sensor noise and feedback error by employing a relatively more conservative method to propagate the intruder uncertainty. The flexibility in resolution actions could compensate for the conservativeness in intruder state propagation.

- Activating a safe collision manoeuvre in a timely manner (enabled by high performance manoeuvres) could reduce the requirement on sensor performance and thus reducing the operational cost of SAA systems.

[1]Note, however, that the method was implemented according to the available description given by Fasano *et al.* [2008] and Carbone *et al.* [2006] and only validated with the testing shown in this thesis. The implementation may not completely reflect the baseline method's effectiveness.

**Limitations** On the other hand, the following limitations of the proposed approach require further research:

- The trajectory planning algorithm will give the best, not necessarily the optimal, trajectory obtained from an iterative process in real time, however, in some very restrictive situations, the resulting trajectory may not be able to meet all the given constraints. Some additional measures may be introduced to prevent the aircraft from entering these restrictive situations, e.g.[Patel & Goulart, 2010], but it is left for future research.

- As there is no analytical solution for the formulated collision avoidance problem, the proposed logic can only be verified statistically rather than analytically. Moreover, it is more time-consuming to statistically verify the proposed approach than other existing approaches. Specifically, the initial prototype is currently able to run at 2Hz, while some other geometric approaches can run at as fast as 50Hz.

- The conservativeness introduced by the uncertainty propagation methods may cause scalability problems. As a collision avoidance logic should be able to handle multiple intruders, the conservative projections of multiple intruders may leave no room for the host aircraft to perform an efficient resolution manoeuvre. Therefore, further analysis will be required to investigate the balance between robustness, conservativeness and scalability.

- The complexity introduced by the trajectory planning algorithm and the additional components of the Trajectory Manager and Trajectory Tracker may increase the difficulties to develop, verify and certify such a system.

**Conclusion of the initial study on the feasibility and potential benefits**

The data and analysis presented in this thesis has shown that real-time trajectory planning for robust collision avoidance with high performance manoeuvres is feasible and potentially beneficial to the development of collision avoidance systems.

From the safety benefit perspective, the proposed approach is expected not only to achieve better performance when the separation is small but also to provide a more timely collision avoidance manoeuvre. From the cost perspective, considering the high-end processing capability is much cheaper than the high-performance sensors, it would be sensible to develop an advanced algorithm to alleviate the stringent sensor performance, and thus reducing the operational cost. Although the development of a safety-critical avionics software with a complex algorithm comes at a high price, the proposed approach is based on a parametric aircraft model that can be easily adapted to a wide range of platforms, implying that it may be able to make a great effort to accomplish something once and for all.

Therefore, this study concludes that it would be worth to further investigate the above limitations and to start looking at the possible issues in the verification and certification process.

## 7.2 Contributions to Knowledge

The key achievements of the work presented in this thesis have contributed to the below two domains as followed:

**Trajectory Planning:**

- **The formulation of an avoidance trajectory planning problem that accommodates the requirements of the flyability of combined manoeuvres and the robustness to the intruder state uncertainty**. Section 5.2 introduces a generic aircraft performance model for the flyability of combined manoeuvres (§5.2.1), an empirical quasi-worse-case projection model for the intruder state uncertainty (§5.2.2) and a cost model for the intruder invisibility in §5.2.4. Although similar models can be found in the existing problem formulations, the combination of them is original.

- **A trajectory planning algorithm that is able to provide an approximate solution to the formulated problem in real time**. Although the algorithm is mainly based on an inverse-dynamic direct method, algorithmic modifications are introduced in §5.3 to improve the real-time performance (via the removal of the virtual domain) and to adapt the algorithm to the collision avoidance problems (via fixing the final time and freeing the other final states). Moreover, the solution space analysis and problem scaling have been carried out in §5.4 to further reduce the computation time. The results presented in §5.5 and §6.2.1 show that the low-level implementation of the algorithm is able to provide a usable avoidance trajectory with a maximum computation time less than 0.5 seconds. The main contribution here is to show that suitable algorithmic modifications can be introduced to make the algorithm to run in real time while providing flyable avoidance trajectories.

**Collision Avoidance Logic:**

- **The introduction of a set of metrics to measure the operational suitability of a collision avoidance logic and the feedback quality during the resolution**

**period**. Sections 4.4.3 and 6.3.3 show how these newly introduced metrics can be used to evaluate the logic's operational suitability and to investigate the effects of the Field of Regard restriction on the logic performance.

- **A novel collision avoidance logic based on the three-layer architecture and the real-time trajectory planning method**. Whilst most existing methods tend to trade off the details in the collision avoidance problems for a reactive-planning logic (i.e. a logic combining the planning, decision making and guidance control into a single process), this thesis proposes to separate the deliberative avoidance trajectory planning from the reactive avoidance manoeuvre execution (§1.3 and §6.2). The separation allows more details to be included in the planning process (§5.2). The novelty of the proposed logic lies mainly in the application of the three-layer architecture and the capability to perform real-time trajectory planning.

- **Demonstration of the feasibility and potential benefits of the proposed collision avoidance logic**. The preliminary analysis presented in §7.1 shows that the proposed collision avoidance logic is feasible and has the potential to reduce the risk of collision while achieving a lower nuisance alert rate than the existing approaches. This provides the motivation for the further development of the proposed logic.

## 7.3 Dissemination of Results

The main deliverables as the results of the work presented in thesis are:

**Software:**

- A modelling, simulation, analysis and evaluation framework that can be used for collision avoidance systems performance evaluation and safety analysis;

- A collision avoidance logic software that is based on the analytical solution of a geometric collision avoidance problem;

- A trajectory planning software capable of planning high performance collision avoidance manoeuvres;
- A proof-of-concept software prototype that can be used to further develop a collision avoidance logic based on the trajectory planning with high performance manoeuvres.

**Publication:**

[1] C. Lai, J. Whidborne. (2012). Automated Return-to-Route Maneuvers for Unmanned Aircraft Systems. In *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*.

[2] C. Lai, J. Whidborne. (2011). Real-Time Trajectory Generation for Collision Avoidance with Obstacle Uncertainty. In *AIAA Guidance, Navigation, and Control Conference*.

[3] C. Lai, M. Lone, P. Thomas, J. Whidborne, and A. Cooke. (2011). On-Board Trajectory Generation for Collision Avoidance in Unmanned Aerial Vehicles. In *2011 IEEE Aerospace Conference*.

[4] C. Lai, J. Whidborne. (2011). Safety-Guaranteed Trajectory Generation for Collision Avoidance in UAVs. In *2011 26th International Conference on Unmanned Air Vehicle Systems*.

[5] C. Lai, J. Whidborne. (2010). Aircraft Route Re-Planning for a Pop-up Obstacle Using a Direct Method. In *2010 UKACC International Conference on Control*.

## 7.4 Recommendations for Future Research

Based on the initial study of the feasibility, merits and drawbacks of the proposed collision avoidance logic, the following possible directions for further research are recommended:

- A mechanism to trigger the automatic collision avoidance manoeuvre should be developed. Although a simple geometric trigger mechanism is used in this

work, coupling the conflict resolution process with the conflict detection process would be able to reduce the false alarms. For instance, Patel & Goulart [2010] proposed and tested three trigger mechanisms based on the capability of real-time trajectory planning.

- Analyses of the proposed logic's robustness to the modelling error, trajectory tracking performance and navigation state error should be performed. On one hand, the logic performance should be dependent on the accuracy for the host aircraft to execute the high-performance avoidance trajectory, while on the other, the Trajectory Tracker has been introduced to compensate for the modelling error and other external disturbance. Therefore, a more detailed robustness analyses, like those carried out by Kochenderfer & Chryssanthacopoulos [2011], would further demonstrate the effectiveness of the proposed logic.

Table 7.1: Summary of the objectives, findings and conclusions of the five simulation experiments in this thesis.

**§4.4.2 Establishment of the baseline performance:**

- The trade-off study (Figure 4.4) shows that the baseline method is able to meet the notional requirement by trading-off the unnecessary alert rate. With a tight alerting threshold (Table 4.5), the logic reduced the risk ratio from 1 to 0.0211 and 0.0656, for the cooperative and non-cooperative cases.
- The baseline method is effective in preventing NMACs.

**§4.4.3 Investigation of the Field-of-Regard restriction effect:**

- Figures 4.6–4.8 quantitatively show that FOR restriction would reduce the quality of the feedback to the collision avoidance logic, increase the risk of collision and make the determination of the moment to issue a CoC more challenging.
- Collision avoidance with non-cooperative sensing is more challenging due to the FOR restriction effects as described above.

**§6.3.1 Safety performance evaluation under ideal conditions:**

- Table 6.3 shows that the proposed method outperformed the baseline method in risk ratio by 50%. Tables 6.4 and 6.5 show that the proposed method also performed better when the distance and time separations are small.
- The proposed method is more effective in preventing NMACs, even when the distance and time separations are small.

**§6.3.2 Robustness analysis to sensor noise:**

- Figure 6.10 shows that, while varying the noise level, the slope of the sensitivity curve of the proposed method is smaller.
- The proposed method is significantly more robust to sensor noise.

**§6.3.3 Evaluation of the FOR restriction effects:**

- The FOR restriction had more impact on the proposed method's safety performance (Figure 6.13). Due to the FOR restriction, the feedback quality to the logic has been decreased even more with the proposed method (Figures 6.14 and 6.15). The proposed method achieved better performance in CoC rate and secondary conflict rate (Figures 6.16 and 6.17). Despite the larger feedback error, the proposed method was able to achieved better safety performance in all surveillance condtions (Figure 6.13).
- The proposed method is significantly more robust to feedback error and more capable of determining a safe terminal state to issue a CoC.

Table 7.2: The main result of the initial feasibility study of the proposed approach—based on a trajectory-planning method for robust collision avoidance with high performance manoeuvres, as described in §1.3.

| Surveillance Conditions | Risk Ratio | |
|---|---|---|
| (under nominal noise, given in Table 3.5) | Baseline[1] | Proposed[2] |
| Cooperative | **0.026**[*] | **0.015**[*] |
| Non-Cooperative[♡] | 0.064 | 0.062 |
| Non-Cooperative (Without Miss Detection)[♠] | **0.035**[*] | **0.033**[*] |

[1] A geometric logic running at 2Hz in a reactive manner.

[2] A trajectory-planning-based logic **running at 2Hz** in a deliberative manner, with a 0.5 seconds computation latency.

[♡] Tight alerting threshold values have been deliberative chosen for the conflict detection mechanism so as to reflect the collision avoidance logic's capability of conflict resolution in emergency situations, see §4.4.2.

[*] A reference risk ratio threshold used in an initial feasibility by Cole *et al.* [2013] is **0.05**.

[♠] The ICAO [2007] Standard Encounter Model is designed for the evaluation of the systems with a cooperative surveillance capability, and it is observed from the results that the intruders in most miss-detection encounters are hard to be detected due the FOR restriction, e.g. the parallel-track situations described in §6.4.3.

[Remark] This is an evaluation using 3800 encounter geometries from the ICAO [2007] Standard Encounter Model (with 1615 original NMAC).

# Appendices

# Appendix A

# Notation

## A.1 Definitions and Typefaces of Mathematical Objects

**With respect to** is denoted by the slash symbol (/), mostly used in a subscript.

**Point** is the mathematical model of a physical object whose spatial extension irrelevant.

It is in Monospace font, e.g. $\mathtt{H}$ denotes the center of mass of the host aircraft.

**Frame of reference** is a rigid body or set of rigidly related points than can be used to establish distances and directions.

It is in Calligraphy font, e.g. $\mathcal{N}$ denotes the navigation frame.

**Euclidean vector** is an abstract geometrical object that has both magnitude and direction. It exists independently of any coordinate system.

It is in bold and italic typeface with an overhead right arrow; and a subscript will be used to give the specific definitions:

1. For a displacement vector: a point with respect to another point, e.g. $\vec{\boldsymbol{r}}_{\mathtt{H/N}}$ denotes the displacement vector of point $\mathtt{H}$ with respect to point $\mathtt{N}$.

2. For a position vector: a point with respect to a frame, e.g. $\vec{\boldsymbol{p}}_{\mathtt{H}/\mathcal{N}}$ denotes the position vector of point $\mathtt{H}$ with respect to (the fixed point $\mathtt{N}$ in) the frame $\mathcal{N}$.

3. For a velocity vector: a point with respect to a frame, e.g. $\vec{\boldsymbol{v}}_{\mathtt{H}/\mathcal{N}}$ denotes the velocity vector of point $\mathtt{H}$ with respect to frame $\mathcal{N}$.

4. For an acceleration vector: a point with respect to a frame, e.g. $\vec{\boldsymbol{a}}_{\mathtt{H}/\mathcal{N}}$ denotes the acceleration vector of point $\mathtt{H}$ with respect to frame $\mathcal{N}$.

**Derivative of an Euclidean Vector** is denoted by the symbol of the Euclidean vector, with a left superscript indicating the frame in which a derivative is taken, and the dot notation indicating a derivative, e.g. ${}^{\mathcal{N}}\dot{\vec{\boldsymbol{p}}}$ denotes the derivative of the position vector, taken in frame $\mathcal{N}$.

**Coordinate system** is a measurement system for locating points in space, set up within a frame of reference.

It is in upper-case sans-serif font, and subscripted by one of the followings for different types of coordinate systems:

1. For the spherical coordinate system: a lower-case sans-serif $\mathsf{s}$, e.g. $\mathsf{B}_{\mathsf{s}}$ denotes the body-fixed spherical coordinate system.

2. For the (Cartesian/rectangle)[1] coordinate system: either a lower-case sans-serif $\mathsf{r}$ or no subscript symbol, e.g. $\mathsf{N}_{\mathsf{r}}$ or $\mathsf{N}$ denotes the navigation (Cartesian) coordinate system.

**Coordinate vector** is a column vector containing the coordinates and associated with a particular coordinate system. An Euclidean vector can be expressed with different coordinate vectors in the different coordinate systems.

It is in bold and italic typeface in square bracket, with a right superscript indicating the corresponding coordinate system, e.g. $\left[\vec{\boldsymbol{p}}_{\mathtt{H}/\mathcal{N}}\right]^{\mathsf{N}}$.

To avoid overloading variable symbol, the notation will be simplified when there is no danger of confusion: to drop the default navigation frame $\mathcal{N}$ and navigation coordinate system $\mathsf{N}$ in the notation; for instance, the position vector $\vec{\boldsymbol{p}}_{\mathtt{H}/\mathcal{N}}$ can be expressed in the navigation coordinate system $\mathsf{N}$ with the following coordinate vectors:

[1]The (Cartesian/rectangle) coordinate system is the default type of coordinate system in this thesis, and the word 'Cartesian' will be dropped when there is no danger of confusion.

$$\left[\vec{\boldsymbol{p}}_{\mathrm{H}/\mathcal{N}}\right]^{\mathsf{N}} = \boldsymbol{p}^{\mathsf{N}}_{\mathrm{H}/\mathcal{N}} \quad \text{or simply} \quad \left[\vec{\boldsymbol{p}}_{\mathrm{H}/\mathcal{N}}\right]^{\mathsf{N}} = \boldsymbol{p}_{\mathrm{H}}. \tag{A.1.1}$$

**Coordinate subscription** is in italic typeface used for subscription. Each coordinate system has its own standard basis, which is a set of mutually orthogonal unit vectors, for instance:

1. for Cartesian coordinate systems: $\langle \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}} \rangle$.
2. for spherical coordinate systems: $\langle \hat{\boldsymbol{r}}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}} \rangle$.
3. for cylindrical coordinate systems: $\langle \hat{\boldsymbol{\rho}}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{h}} \rangle$.
4. for geographic coordinate systems: $\langle \hat{\boldsymbol{N}}, \hat{\boldsymbol{E}}, \hat{\boldsymbol{D}} \rangle$.

Each coordinate is simply a scalar, denoted by the coordinate vector symbol with a subscript indicating the corresponding basis vector, e.g.:

$$\boldsymbol{p}_{\mathrm{H}} = \begin{bmatrix} p_{\mathrm{H},x} \\ p_{\mathrm{H},y} \\ p_{\mathrm{H},z} \end{bmatrix} \quad \text{or} \quad \boldsymbol{p}^{\mathsf{H_s}}_{\mathrm{H}} = \begin{bmatrix} p^{\mathsf{H_s}}_{\mathrm{H},r} \\ p^{\mathsf{H_s}}_{\mathrm{H},\phi} \\ p^{\mathsf{H_s}}_{\mathrm{H},\theta} \end{bmatrix} \quad \text{or} \quad \boldsymbol{p}^{\mathsf{E_c}}_{\mathrm{H}} = \begin{bmatrix} p^{\mathsf{H_c}}_{\mathrm{H},\rho} \\ p^{\mathsf{H_c}}_{\mathrm{H},\phi} \\ p^{\mathsf{H_c}}_{\mathrm{H},h} \end{bmatrix}. \tag{A.1.2}$$

**Vector** is a general column vector as defined in linear algebra.

It is in bold and italic typeface, e.g. $\boldsymbol{x}$ is the state vector.

**Direction Cosine Matrix** is a matrix used to perform the coordinate transformation between different Cartesian coordinate systems, e.g. $\underline{\mathbf{C}}^{\mathsf{H}}_{\mathsf{N}}$ from navigation coordinate system to host coordinate system.

**Coordinate Transformation Function** is a function used to perform the coordinate transformation among the rectangle, spherical and cylindrical coordinate system.

A vector $\vec{\boldsymbol{a}}$, defined in frame $\mathcal{A}$, can be transformed from the rectangular (Cartesian) coordinates to its spherical coordinates with the following coordinate transformation functions $\boldsymbol{T}^{\mathsf{s}}_{\mathsf{r}} : \mathbb{R}^3 \to \mathbb{S}$:

$$\left[\vec{\boldsymbol{a}}\right]^{\mathsf{A_s}} = \boldsymbol{T}^{\mathsf{s}}_{\mathsf{r}} \left( \left[\vec{\boldsymbol{a}}\right]^{\mathsf{A_r}} \right) \tag{A.1.3}$$

where

$$[\vec{\boldsymbol{a}}]^{\mathsf{A_s}} = \begin{bmatrix} r \\ \Phi \\ \Theta \end{bmatrix}; \quad [\vec{\boldsymbol{a}}]^{\mathsf{A_r}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}; \quad \boldsymbol{T}_{\mathsf{r}}^{\mathsf{s}} = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arctan(\frac{y}{x}) \\ -\arcsin(\frac{z}{r}) \end{bmatrix}.$$

# Appendix B

# Aircraft System Model

## B.1 Aircraft Performance Data

Based on the BADA [Nuic, 2012] and the data pack [Cooke, 2008], this section presents the data of the Jetstream 31 aircraft performance model, which have been mentioned in the Chapter 3 and Chapter 5.

Table B.1 summarizes the parameters used for the flight dynamics modelling (§3.4.1) and for the collision avoidance logics (§3.6.2 and §5.2.2). Figure B.1 shows the flight envelope regime used in this work. Figure B.2 shows the maximum load factor look-op table model, which is derived from the aerodynamic force model and the ISA model. Figure B.3 shows the n-V diagram used as the maximum load factor model used in the trajectory planning algorithm, which is obtained from Figure B.2.

## B.2 Verification Examples

This sections presents the example results to verify that the implemented model meet the modelling requirements as specified in Section 3.4. Table B.1 shows the parameter values used for this verification.

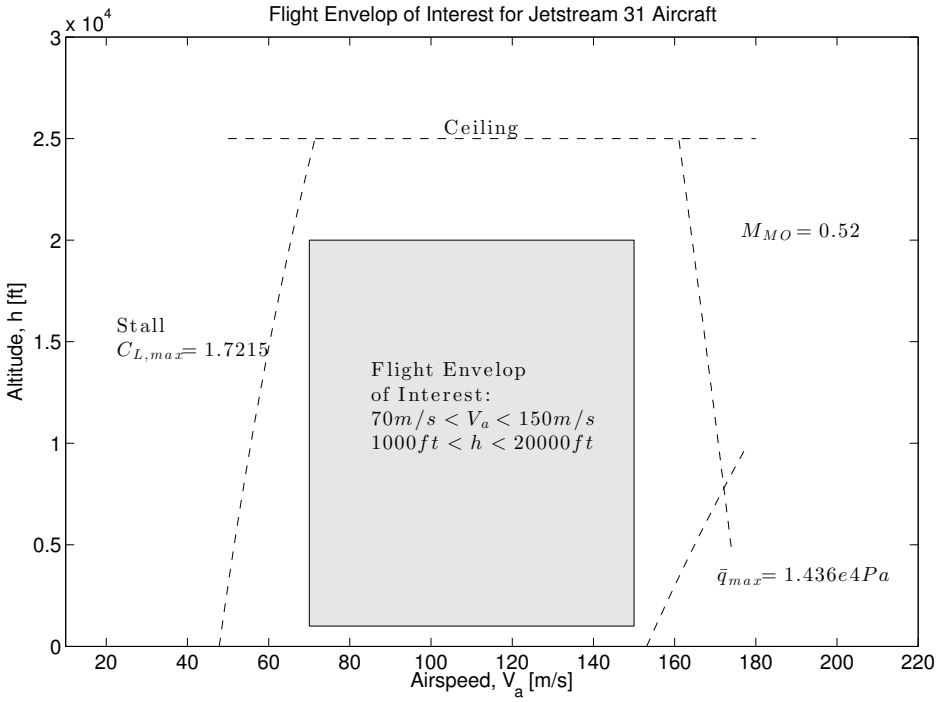



Figure B.1: Flight envelop of the Jetstream 31 aircraft model used in this work.

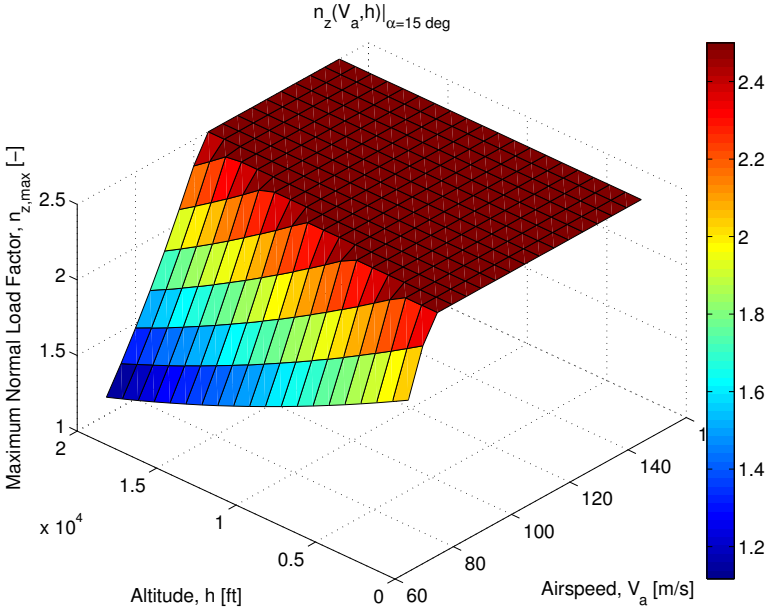



Figure B.2: The maximum load factor look-up table model.

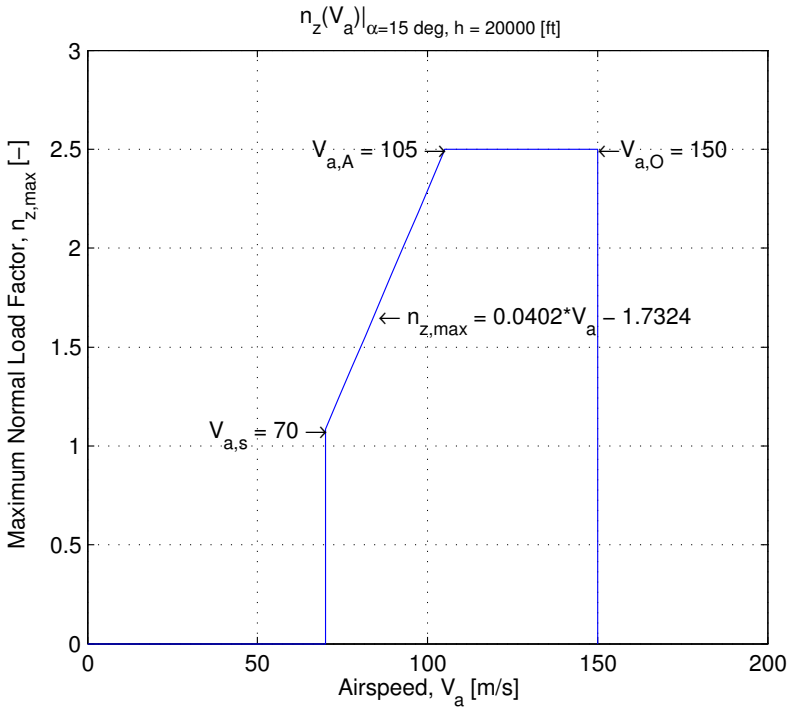



Figure B.3: The n-V diagram used as the maximum load factor model used in the trajectory planning algorithm.

Table B.1: Aircraft performance model parameters.

| Parameter | Symbol | Value | Unit |
|---|---|---|---|
| Reference Mass | $m$ | 6200 | kg |
| Wing area | $S$ | 25.2 | $m^2$ |
| Max. altitude | $h_{max}$ | 25000 | ft |
| Max. airspeed | $V_{a,max}$ | 150 | m/s |
| Min. airspeed | $V_{a,min}$ | 70 | m/s |
| Max. vertical speed | $V_{V,max}$ | 2000 | ft/min |
| Min. vertical speed | $V_{V,min}$ | -2000 | ft/min |
| Max. angle of attack | $\alpha_{max}$ | 15 | deg |
| Min. angle of attack | $\alpha_{min}$ | -3 | deg |
| Max. bank angle | $\mu_{max}$ | 45 | deg |
| Min. bank angle | $\mu_{min}$ | -45 | deg |
| Max. bank angle rate | $\dot{\mu}_{max}$ | 15 | deg/s |
| Min. bank angle rate | $\dot{\mu}_{min}$ | -15 | deg/s |
| Max. structural load factor | $n^{str}_{max}$ | 2.5 | - |
| Max. airspeed for CR | $V^{CR}_{a,max}$ | 119 | m/s |
| Min. airspeed for CR | $V^{CR}_{a,min}$ | 77 | m/s |
| Max. vertical speed for CR | $V^{CR}_{V,max}$ | 1968 | ft/min |
| Min. vertical speed for CR | $V^{CR}_{V,min}$ | -1968 | ft/min |

**Transient and Tracking Performance:** To investigate the model's performance in velocity command tracking, a set of aircraft models, with different time constant settings, is commanded to track the *step* and *ramp* test input signals. Figure B.4 summarises five step responses for the three command channels. It shows that the time constant can be used to model the transient performance of the platform being tested. Figure B.5 summarises the ramp responses of the model with the selected time constants values. As excepted with all controllers with only the proportional term, the ramp responses show constant steady-state errors in all three channels: the commands were able to be tracked with a constant delay, which is determined by the time constant as depicted by the data points.

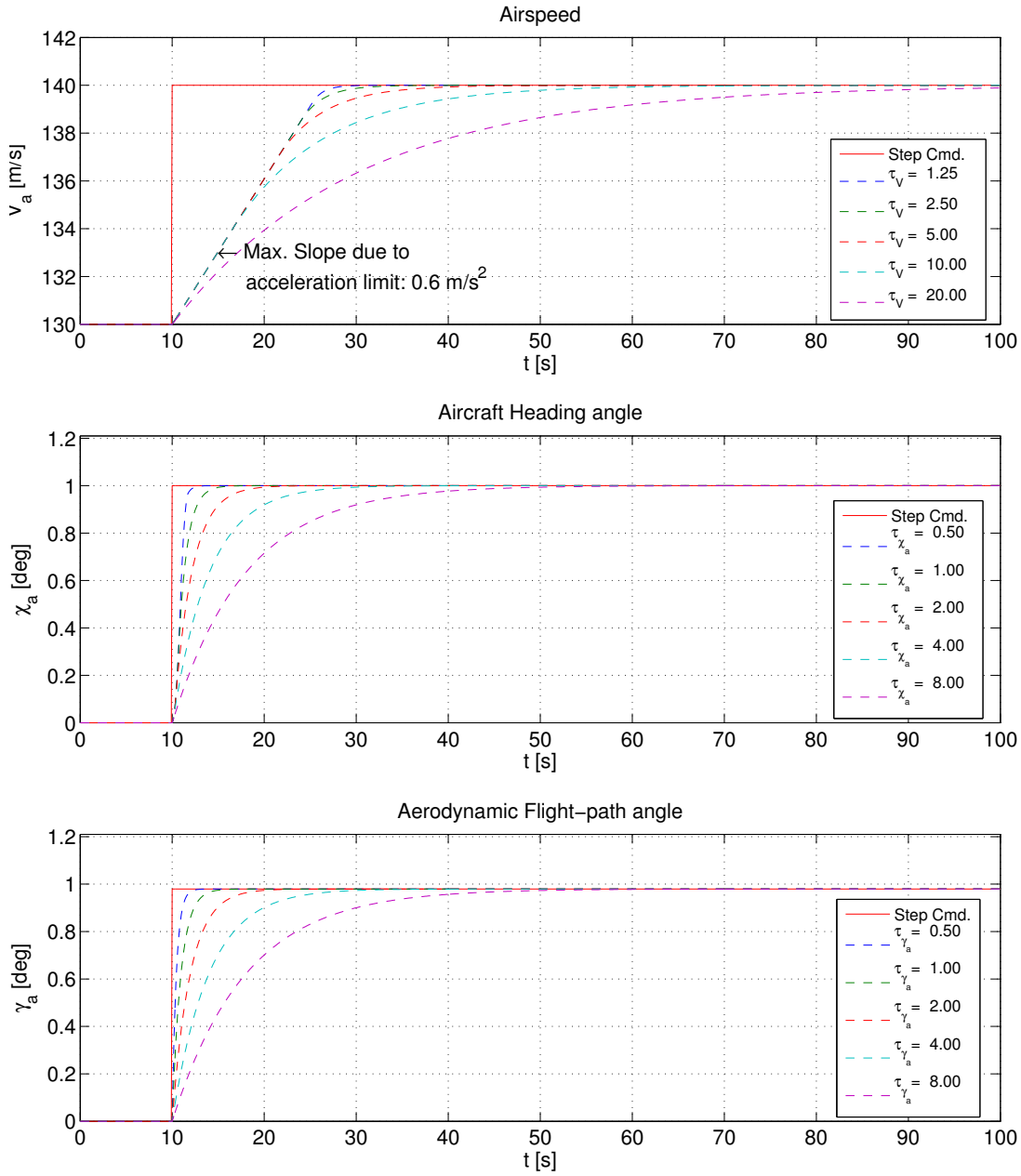



Figure B.4: The step responses of three control channels: Airspeed, Aircraft Heading, and Aerodynamic Flight-Path Angle.

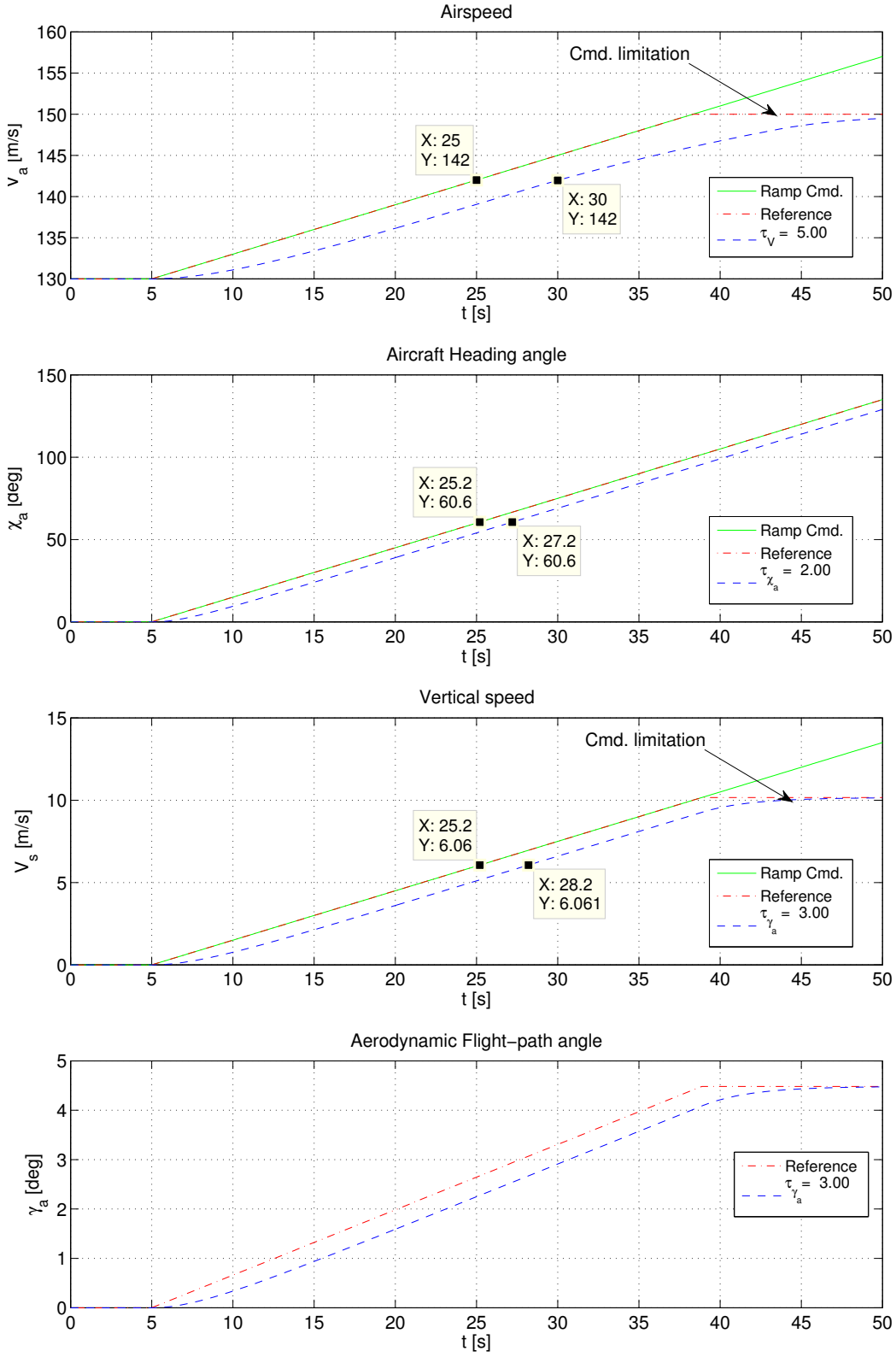



Figure B.5: The ramp responses of the three command channels.

**Attitude Estimation and Performance Limitations:** In order to investigate the correctness of the attitude state and the performance limitation compliance, a series of over-limited manoeuvre commands is used to simulate a flight at the edge of the flight envelop. Figure B.6 shows the time history of the aircraft state and controls. The figure is mainly divided into four parts: Decelerated Only, Turn Only, Climb Only, and Combined (involving all three control channels) manoeuvre.

Firstly, during the pure deceleration phase, the pitch attitude and the angle of attack (denoted by $\theta$ and $\alpha$, respectively) had increased in order to compensate for the reduced dynamic pressure, so as to provide enough lift to maintain the level flight. This can verify the correctness of the aerodynamic force model implementation. Furthermore, the command limits, due to the given aircraft performance limitations, are also verified by the saturation as shown in the $V_a$ and $V_s$ diagrams.

Secondly, given a turn command of 60 degrees, as shown in the actual bank angle $\mu$ history, the model reached it maximum bank of 45 degrees at the maximum rate of 15 degree per second. Furthermore, as illustrated in the actual normal factor $n_z$ history, it is desirable to generate more lift to maintain level flight while at the maximum bank, however, a conservative[1] $n_z$ limit prevents the $n_z$ (and thus $\alpha$) from increasing. Therefore, the vertical speed $V_s$ had to decrease and reached its limit value.

Finally, similar analyses can be found in the climb and combined phases.

[1] *see* the V-n diagram in Appendix A

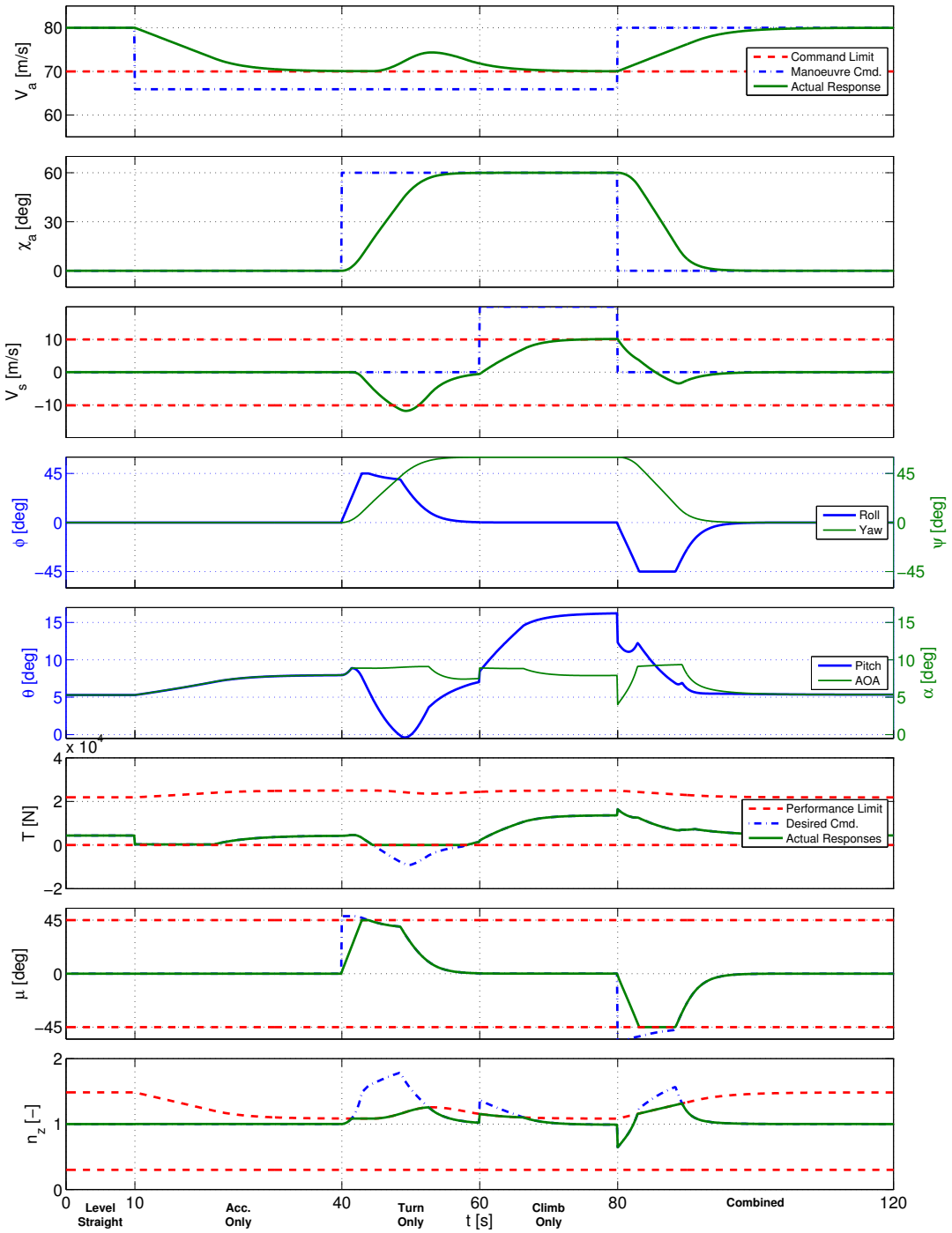



Figure B.6: The responses for a three dimensional manoeuvre with simultaneous changes in three control channels.

# Appendix C

# Trajectory Planning Algorithm Implementation Details

This appendix provides the implementation details of the trajectory planning algorithm and presents the example trajectories of some infeasible solutions obtained from the algorithm, so as to demonstrate the possibility to use these trajectories in the emergency situations.

## C.1 Hooke-Jeeves Algorithm

Based on the description and implementation by [Kelley, 1999, §8.3.1], this section presents two algorithms for the pattern search method used for the work presented in the thesis: Exploratory Move Algorithm C.1 and Hooke-Jeeves Algorithm C.2.

## C.2 Algorithm Parameters

Table C.1 summarizes all the configurable parameters of the trajectory planning algorithm used for the work presented in the thesis.

**Algorithm C.1:** Exploratory Move `hjexplore`

**Input**: $\boldsymbol{x}_B \in \mathbb{R}^N$ ; // Initial *B*ase point vector
$\boldsymbol{x}_C \in \mathbb{R}^N$ ; // Initial *C*entre point vector
$f : \mathbb{R}^N \to \mathbb{R}$ ; // Objective function
$\boldsymbol{h} \in \mathbb{R}^{N_h}, h_{i+1} > h_i$ ; // Search step size vector
$\underline{\mathbf{V}} = [\boldsymbol{v}_1 \cdots \boldsymbol{v}_N] \in \mathbb{R}^{N \times N}$ ; // Search direction matrix

$s_f \leftarrow 0$ ; // Initialization
$f_B \leftarrow f(\boldsymbol{x}_B)$ ; $\sigma \leftarrow 1$ ; // Evaluate *B*ase point
$\boldsymbol{x}_{best} \leftarrow \boldsymbol{x}_B$; $f_{best} \leftarrow f_B$; // Set *best* point to *B*ase point
$\boldsymbol{x}_t = \boldsymbol{x}_C$ ; // Set *t*emporary point to *C*entre point
**for** $j \leftarrow 1$ **to** $N$ **do** // Perform exploration for each direction
  $p \leftarrow \boldsymbol{x}_t + h\boldsymbol{v}_j$; $f_t \leftarrow f(p)$ ; $\sigma \leftarrow \sigma + 1$ ;
  **if** $f_t \geq f_B$ **then**
    $p = \boldsymbol{x}_t - h\boldsymbol{v}_j$; $f_t \leftarrow f(p)$ ; $\sigma \leftarrow \sigma + 1$;
  **end**
  **if** $f_t < f_B$ **then**
    $s_f \leftarrow 1$; $\boldsymbol{x}_t \leftarrow p$; $f_B = f_t$ ;
  **end**
  **if** $s_f = 1$ **then**
    $\boldsymbol{x}_{best} = \boldsymbol{x}_t$; $f_{best} \leftarrow f_t$ ;
  **end**
**end**

**Algorithm C.2:** Hooke-Jeeves Algorithm `hjsearch`

**Input**: $\boldsymbol{x}_0 \in \mathbb{R}^N$ // Initial point vector
$f : \mathbb{R}^N \to \mathbb{R}$ // Objective function
$\boldsymbol{h} \in \mathbb{R}^{N_h}, h_{i+1} > h_i$ // Search step size vector
$\underline{\mathbf{V}} = [\boldsymbol{v}_1 \cdots \boldsymbol{v}_N] \in \mathbb{R}^{N \times N}$ // Search direction matrix
$\sigma_{max}$ // Maximum function evaluations
$\epsilon_f$ // Stopping criterion in objective tolerance
$f_{tar}$ // Stopping criterion in objective target

**Output**: $\boldsymbol{x}$ // Best point after the search
$flag \in \{0, 1, 2\}$ // Search status

$i \leftarrow 1$; $\boldsymbol{x} \leftarrow \boldsymbol{x}_0$; $f_0 \leftarrow f(\boldsymbol{x}_0)$; $\sigma \leftarrow 1$ // Initialization
**while** $i \leq N_h$ *and* $\sigma \leq \sigma_{max}$ **do**
  $\left(\boldsymbol{x}, f_i, s_f, \sigma_{exp}\right) \leftarrow$ `hjsearch`$(\boldsymbol{x}, f, h_i, \underline{\mathbf{V}}, \sigma, \sigma_{max})$
  $\sigma \leftarrow \sigma + \sigma_{exp}$
  **if** $f_i \leq f_{tar}$ **then**
    $flag \leftarrow 2$; **return** // Target-optimal
  **else if** $\left(s_f = 0\right) \wedge (i = N_h) \wedge \left(|f_i - f_{i-1}| \leq \epsilon_f\right)$ **then**
    $flag \leftarrow 1$; **return** // Tolerance-optimal
  **else**
    $flag \leftarrow 0$; **return** // Best-within-budget
  **end**
  $i \leftarrow i + 1$
**end**
// Search with one step size
**Hooke-Jeeves Search** $\left(\boldsymbol{x}, f_{best}, s_f, \sigma\right) \leftarrow$ `hjsearch`$(\boldsymbol{x}_B, f, h, \underline{\mathbf{V}}, \sigma_0, \sigma_{max})$
  $\boldsymbol{x}_C \leftarrow \boldsymbol{x}_B$; $\sigma \leftarrow \sigma_0$
  $\left(\boldsymbol{x}, f_{best}, s_f, \sigma_{exp}\right) \leftarrow$ `hjexplore`$(\boldsymbol{x}_B, \boldsymbol{x}_C, f, h, \underline{\mathbf{V}})$; $\sigma \leftarrow \sigma + \sigma_{exp}$
  **while** $\left(s_f = 1\right) \wedge (\sigma < \sigma_{max})$ **do**
    $\boldsymbol{d} \leftarrow \boldsymbol{x} - \boldsymbol{x}_B$; $\boldsymbol{x}_B \leftarrow \boldsymbol{x}$; $\boldsymbol{x}_C \leftarrow \boldsymbol{x} + \boldsymbol{d}$ // Pattern move
    $\left(\boldsymbol{x}, f_{best}, s_f, \sigma_{exp}\right) \leftarrow$ `hjexplore`$(\boldsymbol{x}_B, \boldsymbol{x}_C, f, h, \underline{\mathbf{V}})$; $\sigma \leftarrow \sigma + \sigma_{exp}$
    **if** $s_f = 0$ **then** // Pattern move fails
      $\left(\boldsymbol{x}, f_{best}, s_f, \sigma_{exp}\right) \leftarrow$ `hjexplore`$(\boldsymbol{x}_B, \boldsymbol{x}_B, f, h, \underline{\mathbf{V}})$
      $\sigma \leftarrow \sigma + \sigma_{exp}$
    **end**
  **end**

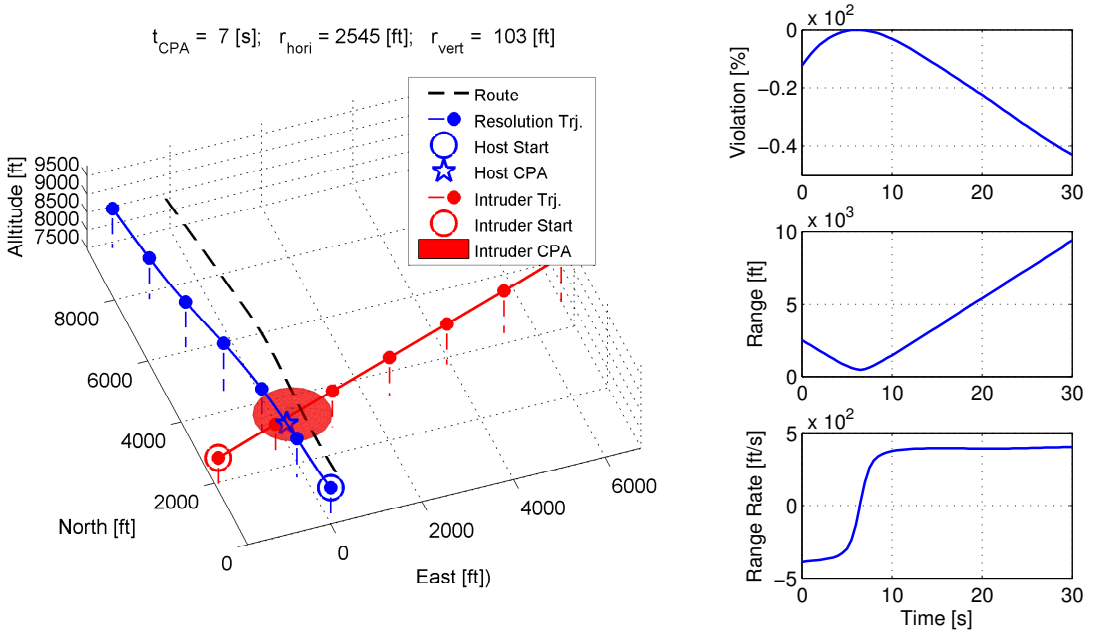
Table C.1: Trajectory Planning Algorithm Parameters.

| Group | Parameter | Symbol | Value | Unit |
|---|---|---|---|---|
| Protected Volume | Growth Factor (5.2.11) | $g$ | 2 | - |
| | Time Constant (5.2.11) | $\tau$ | 30 | s |
| | Vertical Separation (5.2.9) | $H$ | 100 | ft |
| | Horizontal Separation (5.2.9) | $R$ | 500 | ft |
| | Scale Factor (5.2.9) | $\lambda$ | 1 | - |
| Solution Space | Max. Final Position Azimuth (5.4.5) | $\Phi_{pf,max}$ | 45 | deg |
| | Min. Final Position Azimuth (5.4.5) | $\Phi_{pf,min}$ | -45 | deg |
| | Extreme Jerk Value (5.4.6) | $\dddot{p}_{max}$ | 50 | $m^4/s$ |
| | Extrem Ground Track Change (5.4.9) | $\Delta\chi$ | 90 | deg |
| Hooke-Jeeves | Search Step Size Vector §C.1 | $\boldsymbol{h}$ | $1.3^{-i}$, $i = 0,\ldots,20$ | - |
| | Search Direction Matrix §C.1 | $\underline{\mathbf{V}}$ | see C.2.1 | - |
| | Objective Tolerance §C.1 | $\epsilon_f$ | $1 \times 10^{-8}$ | - |
| | Objective Target §C.1 | $f_{tar}$ | 1 | - |
| | Maximum Function Evaluation §C.1 | $\sigma_{max}$ | 3000 | - |

$$\underline{\mathbf{V}} = \mathbf{diag}([0.01 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.001 \quad 0.001 \quad 0.001 \quad 0.001 \quad 0.001 \quad 0.001 \quad 0.01 \quad 0.01 \quad 0.01 \quad 0.001 \quad 0.001]^T) \tag{C.2.1}$$

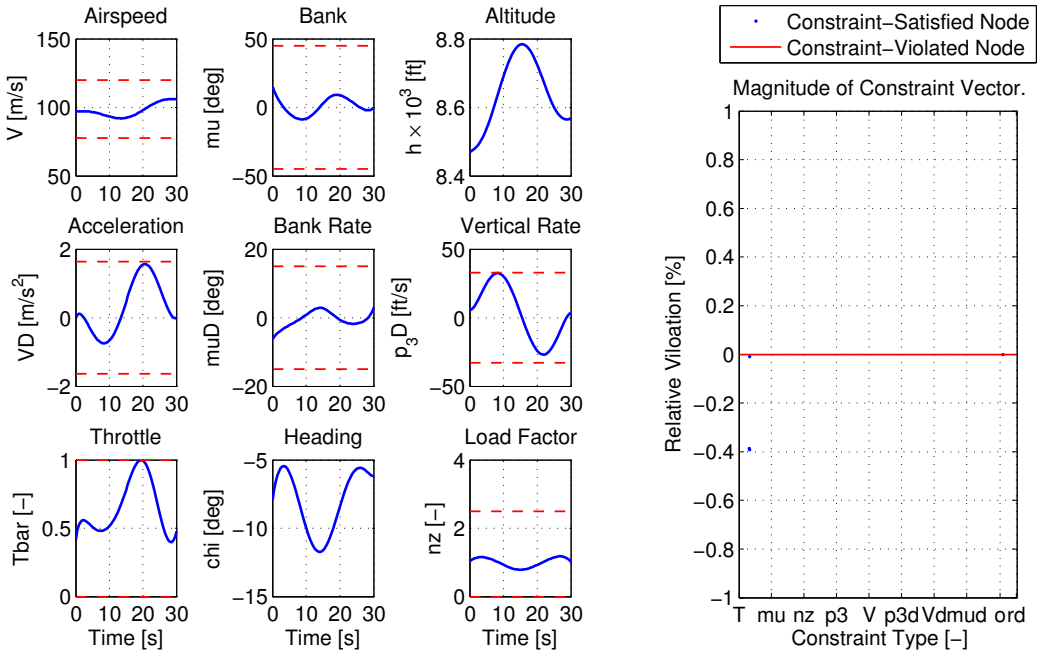
## C.3 Example Trajectories for the Infeasible Solutions

This section summarise the examples of acceptable, infeasible-obstacle, infeasible-performance, and both-infeasible trajectories, in Figures C.2-C.5. In each figure, the resulting miss distance and constraint violations of the each infeasible-solution trajectory are highlighted. Although no feasible solutions could be found, within the maximum number of function evaluations, for these situations, it can be seen that the aircraft performance required by these resulting trajectories' are very close to the specified limit and their miss distances are all larger than the required separation.




(a) Resolution trajectory and the initial linear time-to-CPA, horizontal and vertical range.

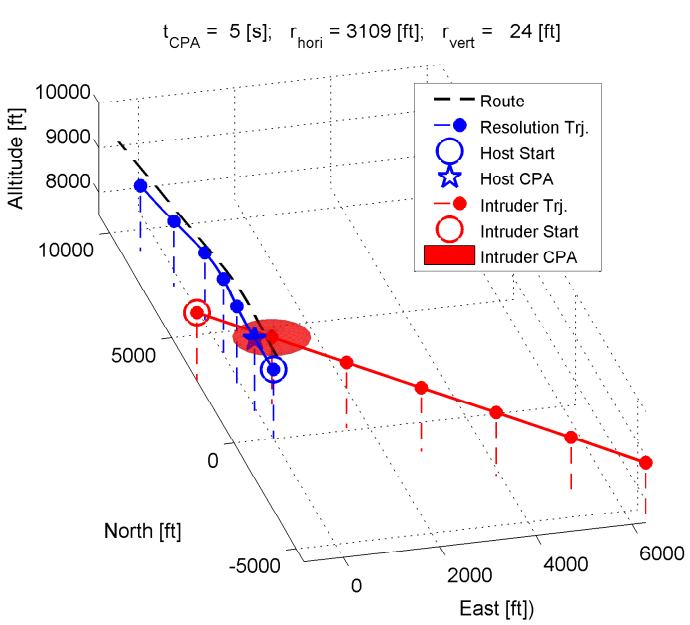
(b) Obstacle constraint and relative range.



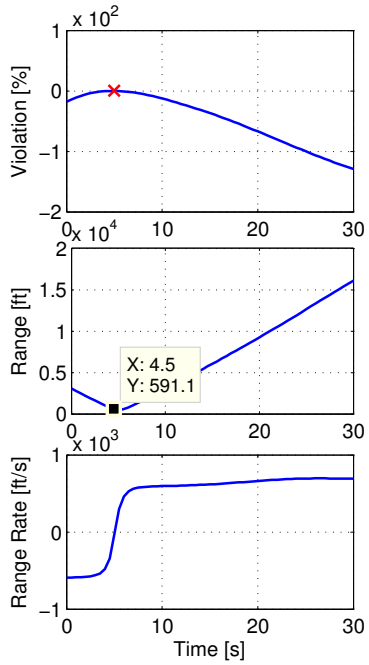

(c) Aircraft state and control trajectories and their performance limits.

(d) Constraint vector evaluated at all the collocation points (nodes).

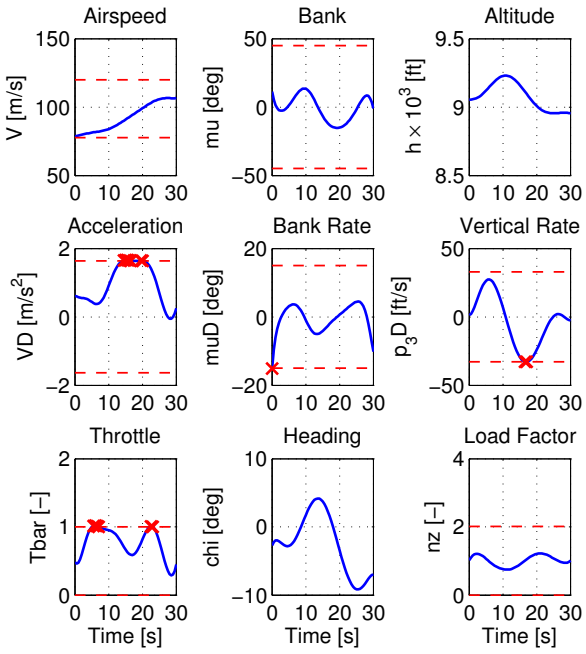
Figure C.1: A feasible trajectory example.



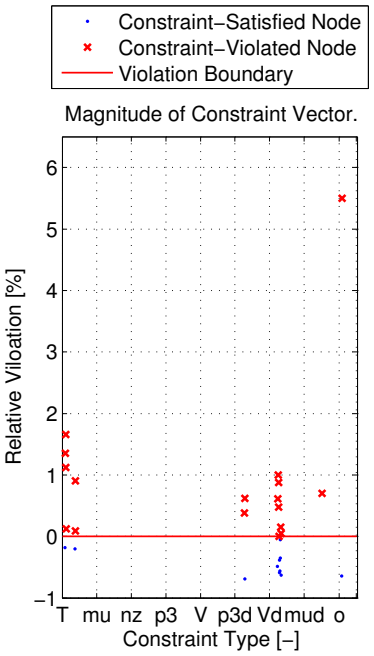

(a) Resolution trajectory and the initial linear time-to-CPA, horizontal and vertical range.




(b) Obstacle constraint and relative range.

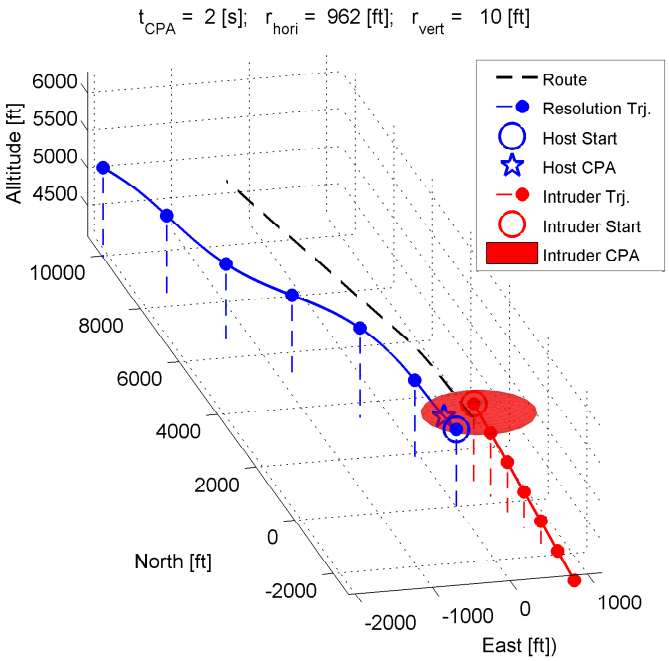



(c) Aircraft state and control trajectories and their performance limits.

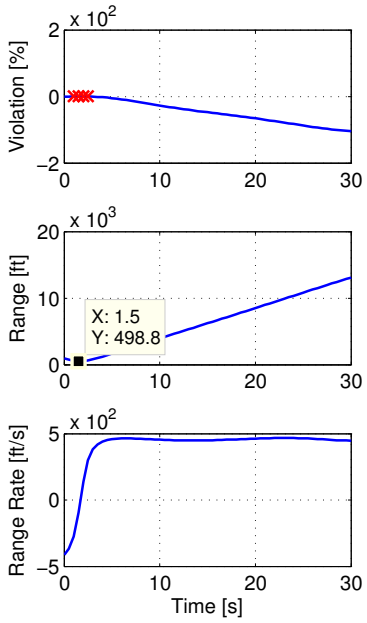



(d) Constraint vector evaluated at all the collocation points (nodes).

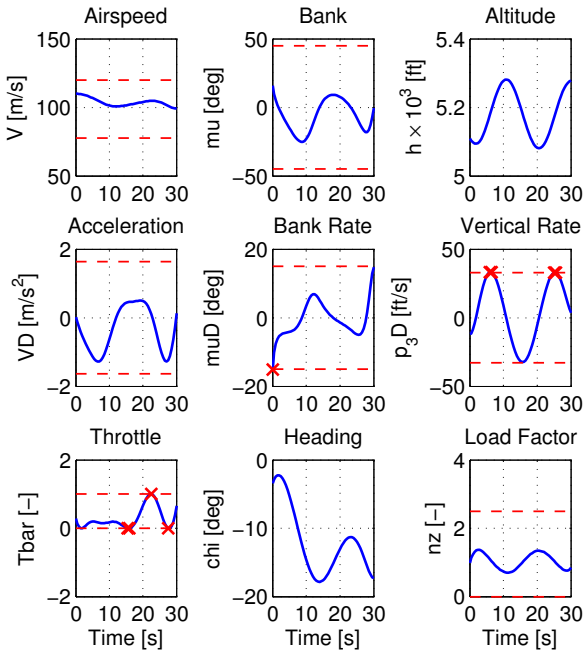
Figure C.2: An acceptable trajectory example. Most performance violations are within 2% of their limits; and at its maximum constraint violation, the miss distance is 591 ft, that is 91 ft further than the required separation.



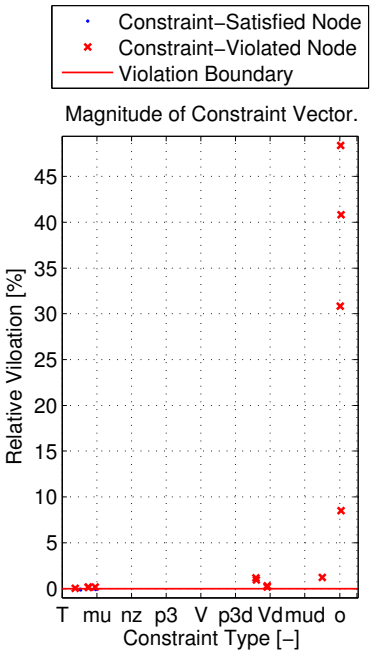

(a) Resolution trajectory and the initial linear time-to-CPA, horizontal and vertical range.




(b) Obstacle constraint and relative range.

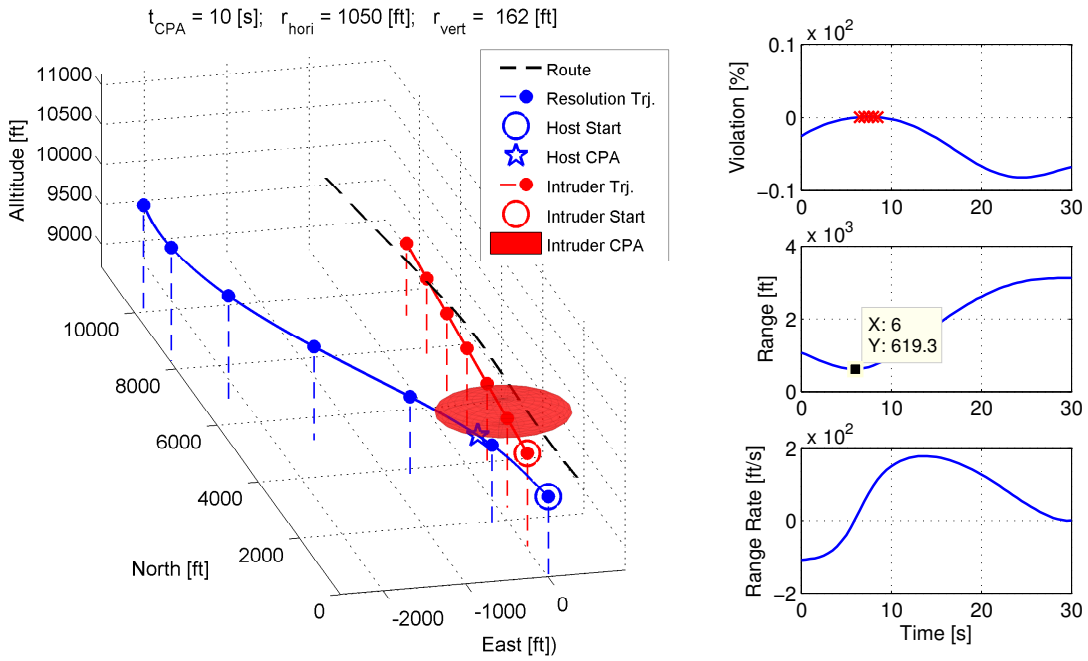



(c) Aircraft state and control trajectories and their performance limits.



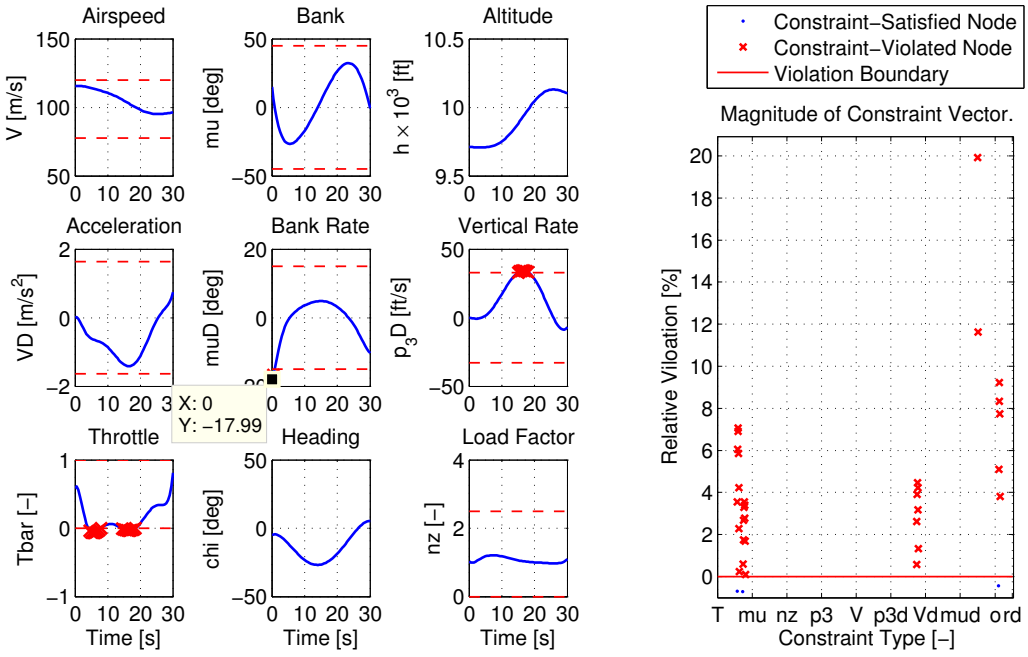

(d) Constraint vector evaluated at all the collocation points (nodes).

Figure C.3: An infeasible-obstacle trajectory example. Despite the 49% of obstacle constraint violation, the estimated miss distance and the required performance show the trajectory's applicability.




(a) Resolution trajectory and the initial linear time-to-CPA, horizontal and vertical range.

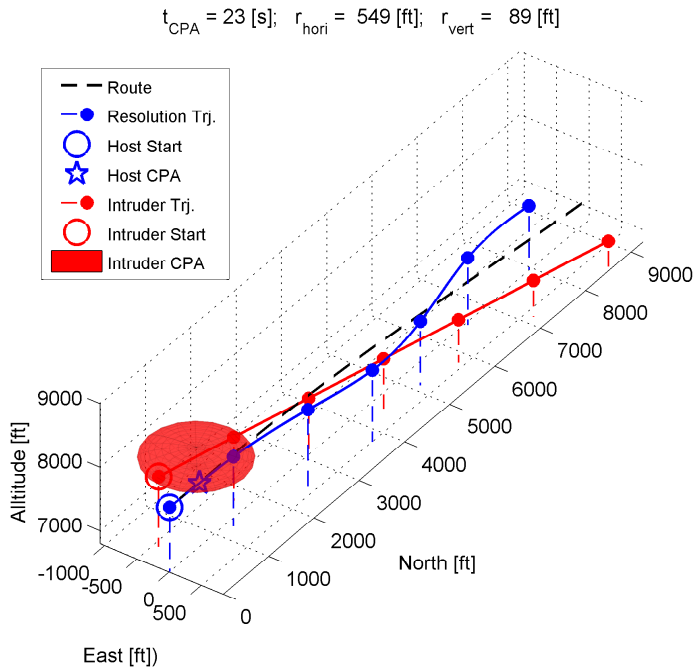
(b) Obstacle constraint and relative range.



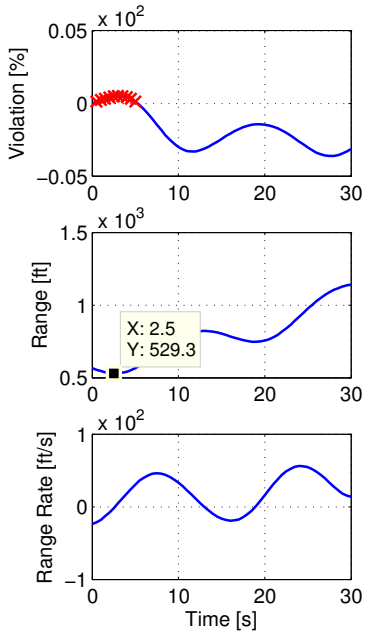

(c) Aircraft state and control trajectories and their performance limits.

(d) Constraint vector evaluated at all the collocation points (nodes).

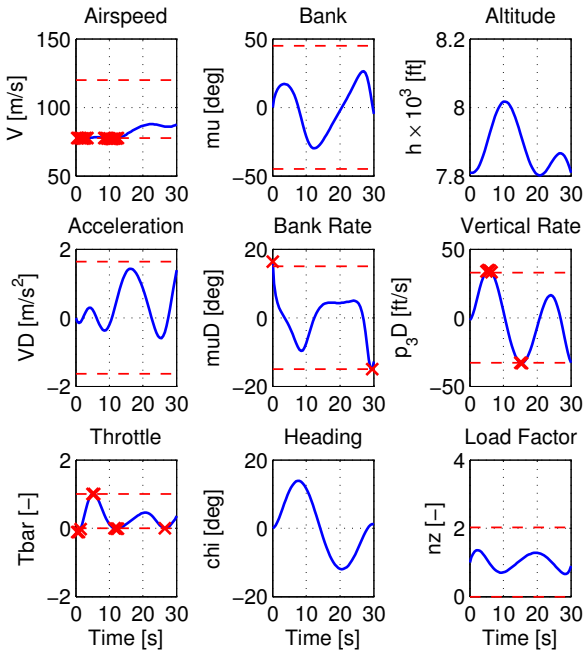
Figure C.4: An infeasible-performance trajectory example.



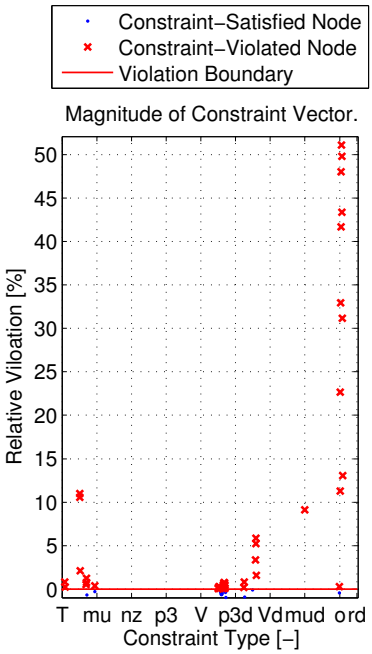

(a) Resolution trajectory and the initial linear time-to-CPA, horizontal and vertical range.




(b) Obstacle constraint and relative range.




(c) Aircraft state and control trajectories and their performance limits.




(d) Constraint vector evaluated at all the collocation points (nodes).

Figure C.5: A both-infeasible trajectory example.

This page intentionally contains only this sentence.

# References


[Angelov, 2012] Angelov, P., ed. (2012). *Sense and Avoid in UAS: Research and Applications*. Aerospace, Wiley, Hoboken, NJ. 18

[Angelov *et al.*, 2008] Angelov, P., Bocaniala, C.D., Xideas, C., Patchett, C., Ansell, D., Everett, M. & Leng, G. (2008). A passive approach to autonomous collision detection and avoidance in uninhabited aerial systems. 23

[Arino *et al.*, 2002] Arino, T., Carpenter, K., Chabert, S., Hutchinson, H., Miquel, T., Raynaud, B., Rigotii, K. & Vallauri, E. (2002). WP-1—Studies on the safety of ACAS II in Europe. Tech. Rep. ACASA/WP-1.8/210D, EUROCONTROL. 30

[Bai & Hsu, 2011] Bai, H. & Hsu, D. (2011). Unmanned aircraft collision avoidance using continuous-state POMDPs. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA. 4

[Bakker, 2009] Bakker, B. (2009). EUROCONTROL guidance material for short term conflict alert. Tech. Rep. EUROCONTROL-GUID-123, EUROCONTROL. 13, 21

[Barhydt *et al.*, 2003] Barhydt, R., Eischeid, T., Palmer, M. & Wing, D. (2003). Use of a prototype airborne separation assurance system for resolving near-term conflicts during autonomous aircraft operations. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, TX. 13, 21

[Basset *et al.*, 2010] Basset, G., Xu, Y. & Yakimenko, O.A. (2010). Computing short-time aircraft maneuvers using direct methods. *Journal of Computer and Systems Sciences International*, **49**, 481–513. 26, 27

[Bayen *et al.*, 2003] Bayen, A., Santhanam, S., Michell, I. & Tomlin, C. (2003). A differential game formulation of alert levels in etms data for high altitude traffic. In *AIAA Guidance, Navigaion, and Coontrol Conference*, Texas. 24, 138

[Berry *et al.*, 2010] Berry, A., Howitt, J., Postlethwaite, I. & Gu, D. (2010). Enabling the operation of multiple micro-air-vehicles in increasingly complex obstacle-rich environments. In *AIAA Infotech@ Aerospace Conference*, Atlanta, Georgia. 22

[Betts, 1998] Betts, J.T. (1998). Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, **21**, 193–207. 26, 27

[Betts, 2010] Betts, J.T. (2010). *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, Philadelphia. 26, 108

[Bollino & Lewis, 2008] Bollino, K. & Lewis, L.R. (2008). Collision-free multi-UAV optimal path planning and cooperative control for tactical applications. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Reston, VA. 26

[Boskovic *et al.*, 2013] Boskovic, J., Jackson, J.A. & Mehra, R. (2013). Sensor and tracker requirements development for sense and avoid systems for unmanned aerial vehicles. In *AIAA Modeling and Simulation Technologies Conference*, Boston, MA. 18

[Buchanan, 2012] Buchanan, R. (2012). Detect and avoid. ASTRAEA National Conference 2012. 2

[CAA, 2012] CAA (2012). *Unmanned Aircraft System Operations in UK Airspace - Guidance*. TSO on behalf of the UK Civil Aviation Authority, Norwich. 1

[Carbone *et al.*, 2006] Carbone, C., Ciniglio, U., Corraro, F. & Luongo, S. (2006). A novel 3D geometric algorithm for aircraft autonomous collision avoidance. In *2006 45th IEEE Conference on Decision and Control*, San Diego, CA. xv, 7, 24, 60, 61, 63, 64, 144

[Chakravarthy & Ghose, 1998] Chakravarthy, A. & Ghose, D. (1998). Obstacle avoidance in a dynamic environment: a collision cone approach. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, **28**, 562–574. 7

[Chen *et al.*, 2011] Chen, R.H., Gevorkian, A., Fung, A., Chen, W.Z. & Raska, V. (2011). Multi-sensor data integration for autonomous sense and avoid. In *AIAA Infotech@ Aerospace Technical Conference*, St. Louis, MO. xxi, 2, 18, 19

[Chen *et al.*, 2009] Chen, W.Z., Wong, L., Kay, J. & Raska, V.M. (2009). Autonomous sense and avoid (SAA) for unmanned air systems (UAS). In *SCI-202 Symposium on "Intelligent Uninhabited Vehicle Guidance Systems"*, Universität der Bundeswehr, München, Germany. 3, 4, 54

[Cho *et al.*, 2012] Cho, S., Huh, S., Shim, D.H. & Choi, H.S. (2012). Vision-based detection and tracking of airborne obstacles in a cluttered environment. *Journal of Intelligent & Robotic Systems*, **69**, 475–488. 23

[Clarkson, 2012] Clarkson, D. (2012). Scenario assessment. Tech. Rep. MIDCAS-T-0128, MIDCAS. 16

[Cole *et al.*, 2013] Cole, R., Kochenderfer, M.J., Weibel, R., Edwards, M.W.M., Griffith, J.D. & Olson, W. (2013). Fielding a sense and avoid capability for unmanned aircraft systems: Policy, standards, technology, and safety modeling. *Air Traffic Control Quarterly*, **21**, 5–27. 15, 79, 81, 144, 152

[Cooke, 2008] Cooke, A.K. (2008). Data pack for the Jetstream 31. Lecture Note FDP Assignment, Cranfield University. 7, 162

[Dixon, 2011] Dixon, R. (2011). BAES S&A interface control document (ICD). Tech. Rep. BAES/AS&FC/W/7Z1/ID/000763, BAE Systems, Warton. xiv, 18, 29

[Drozdowski & Dean, 2010] Drozdowski, S. & Dean, G. (2010). Unmanned aircraft systems - ATM collision avoidance requirements. Tech. Rep. CND/CoE/CNS/09-156, EUROCONTROL, Brussels. 1, 12

[Drury *et al.*, 2010] Drury, R., Tsourdos, A. & Cooke, A. (2010). Real-time trajectory generation: Improving the optimality and speed of an inverse dynamics method. In *IEEE Aerospace Conference*, Big Sky, MT. 26

[Drury, 2010] Drury, R.G. (2010). *Trajectory Generation for Autonomous Unmanned Aircraft Using Inverse Dynamics*. Ph.D. thesis, Cranfield University. 27, 111

[Ducard, 2009] Ducard, G.J.J. (2009). *Fault-tolerant Flight Control and Guidance Systems: Practical Methods for Small Unmanned Aerial Vehicles*. Springer. 41

[Edwards *et al.*, 2009] Edwards, M., Kochenderfer, M.J., Kuchar, J.K. & Espindle, L.P. (2009). Encounter models for unconventional aircraft version 1.0. Project Report ATC-348, MIT Lincoln Laboratory. 31

[Edwards, 2012] Edwards, M.W.M. (2012). A safety driven approach to the development of an airborne sense and avoid system. In *AIAA Infotech@Aerospace Conference*, Garden Grove, California. 2, 15, 16, 88, 132

[F38 Committee, 2007] F38 Committee (2007). Specification for design and performance of an airborne sense-and-avoid system. Tech. Rep. F2411-07, ASTM International. 55, 87

[FAA, 2008] FAA (2008). *Interim Operational Approval Guidance 08-01: Unmanned Aircraft Systems Operations in the U. S. National Airspace System*. 1

[FAA-Sponsored Workshop, 2009] FAA-Sponsored Workshop (2009). Sense and avoid (SAA) for unmanned aircraft systems (UAS). Workshop final report, FAA. xiii, 17, 18

[Farjon & Sellem-Delmar, 2012] Farjon, J. & Sellem-Delmar, S. (2012). MIDCAS concept of operations (CONOPS). Tech. Rep. MIDCAS-T-0017, MIDCAS. 16, 17

[Fasano, 2008] Fasano, G. (2008). *Multisensor based Fully Autonomous Non-Cooperative Collision Avoidance System for UAVs*. Ph.D. thesis, Università degli Studi di Napoli Federico II. 57

[Fasano *et al.*, 2008] Fasano, G., Accardo, D., Moccia, A., Carbone, C., Ciniglio, U., Corraro, F. & Luongo, S. (2008). Multi-sensor-based fully autonomous non-cooperative collision avoidance system for unmanned air vehicles. *Journal of Aerospace Computing, Information, Communication*, **5**, 338–360. xv, 23, 54, 60, 144

[Fasano *et al.*, 2009] Fasano, G., Accardo, D., Moccia, A. & Rispoli, A. (2009). Flight test results for a multi sensor obstacle detection and tracking system for sense and avoid applications. In *AIAA Infotech and Aerospace*, vol. 2009, Seattle, Washington. 19

[Fisch, 2011] Fisch, F. (2011). *Development of a Framework for the Solution of High-Fidelity Trajectory Optimization Problems and Bilevel Optimal Control Problems*. Ph.D. thesis, Technische Universität München. 46

[Fliess *et al.*, 1995] Fliess, M., Levine, J., Martin, P. & Rouchon, P. (1995). Flatness and defect of nonlinear systems: Introductory theory and examples. *International Journal of Control*, **61**, 1327–1361. 98

[Flores, 2007] Flores, M.E. (2007). *Real-Time Trajectory Generation for Constrained Nonlinear Dynamical Systems Using Non-Uniform Rational B-spline Basis Functions*. Ph.D. thesis, California Institute of Technology. 26

[Gat, 1998] Gat, E. (1998). On three-layer architectures. In *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, 195–210, AAAI Press. 4, 121

[George, 2012] George, S. (2012). Scope and intended function of systems and equipment providing sense and Avoid/Detect and avoid (SAA/DAA) capability for unmanned aircraft systems (UAS). Tech. Rep. UASSG/9-SN/4, ICAO. xiii, 17, 18

[Gill *et al.*, 2008] Gill, P.E., Murray, W. & Saunders, M.A. (2008). User's guide for SNOPT version 7: Software for large-scale nonlinear programming. User manual, Stanford Business Software Inc. 111

[Goerzen *et al.*, 2010] Goerzen, C., Kong, Z. & Mettler, B. (2010). A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems*, **57**, 65–100. 26

[Griffin & Kolda, 2007] Griffin, J.D. & Kolda, T.G. (2007). Nonlinearly-constrained optimization using asynchronous parallel generating set search. Tech. Rep. SAND2007-3257, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, CA. 110

[Griffin & Kolda, 2010] Griffin, J.D. & Kolda, T.G. (2010). Nonlinearly constrained optimization using heuristic penalty methods and asynchronous parallel generating set search. *Applied Mathematics Research eXpress*, **2010**, 36–62. 110

[Griffith & Lee, 2011] Griffith, J.D. & Lee, S. (2011). Environment modeling for sense and avoid sensor safety assessment. In *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th*, IEEE. 16

[Griffith *et al.*, 2008] Griffith, J.D., Kochenderfer, M.J. & Kuchar, J.K. (2008). Electro-optical system analysis for sense and avoid. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Honolulu, Hawaii. 18

[Hoekstra, 2002] Hoekstra, J.M. (2002). Free flight with airborne separation assurance. Tech. Rep. NLR-TP-2002-170, NLR. 13

[Hoffren & Sailaranta, 2001] Hoffren, J. & Sailaranta, T. (2001). Maneuver autopilot for realistic performance model simulations. In *Proceedings of the 2001 AIAA Modeling and Simulation Technologies Conference and Exhibit*, AIAA. 46

[Holland *et al.*, 2013] Holland, J.E., Kochenderfer, M.J. & Olson, W.A. (2013). Optimizing the next generation collision avoidance system for safe, suitable, and acceptable operational performance. In *Tenth USA/Europe Air Traffic Management Research and Development Seminar*, Chicago, Illinois. 72, 79

[Hooke & Jeeves, 1961] Hooke, R. & Jeeves, T.A. (1961). " Direct search" solution of numerical and statistical problems. *J. ACM*, **8**, 212–229. 111

[Huerta, 2013] Huerta, M.P. (2013). *Integration of Civil Unmanned Aircrat Systems (UAS) in the National Airspace System (NAS) Roadmap*. FAA, Washington, DC, 1st edn. 1

[Hull, 1997] Hull, D.G. (1997). Conversion of optimal control problems into parameter optimization problems. *Journal of Guidance, Control, and Dynamics*, **20**, 57–60. 27

[Hull, 2010] Hull, D.G. (2010). *Fundamentals of Airplane Flight Mechanics*. Springer, Berlin; New York. 41, 43

[Hutchings *et al.*, 2007] Hutchings, T., Jeffryes, S. & Farmer, S. (2007). Architecting UAV sense and avoid systems. In *Autonomous Systems, 2007 Institution of Engineering and Technology Conference on*, 1–8. xiv, 15, 18, 20

[Hutchinson & Drozdowski, 2007] Hutchinson, H. & Drozdowski, S. (2007). Report on RA downlink contingency tree study. Tech. Rep. QinetiQ/D&T/C&IS/ADC/RADLCT/13/03, EUROCONTROL. 16

[ICAO, 2005a] ICAO (2005a). *Annex 2 to the Convention on International Civil Aviation: Rules of the Air*. ICAO, Montréal, Québec, tenth edn. 1, 14

[ICAO, 2005b] ICAO (2005b). *Global Air Traffic Management Operational Concept*. ICAO, Montréal, Québec, 1st edn. 12

[ICAO, 2007] ICAO (2007). *Annex 10 to the Convention on International Civil Aviation: Aeornautical Telecommunications*, vol. IV Surveillance and Collision Avoidance Systems. ICAO, Montréal, Québec, 4th edn. 7, 16, 21, 24, 30, 31, 32, 33, 35, 36, 60, 73, 79, 80, 88, 141, 152

[Jardin, 2003] Jardin, M.R. (2003). *Toward Real-Time En Route Air Traffic Control Optimization*. Ph.D. thesis, Stanford University. 21

[Johansen & Fossen, 2013] Johansen, T.A. & Fossen, T.I. (2013). Control allocation—A survey. *Automatica*, **49**, 1087–1103. 47

[Kelley, 1999] Kelley, C.T. (1999). *Iterative Methods for Optimization*, vol. 18. SIAM, Philadelphia. 111, 171

[Kochenderfer & Chryssanthacopoulos, 2011] Kochenderfer, M.J. & Chryssanthacopoulos, J.P. (2011). Robust airborne collision avoidance through dynamic programming. Project Report ATC-371, MIT Lincoln Laboratory. 150

[Kochenderfer *et al.*, 2008a] Kochenderfer, M.J., Espindle, L.P., Kuchar, J.K. & Griffith, J.D. (2008a). A comprehensive aircraft encounter model of the national airspace system. *Lincoln Laboratory Journal*, **17**, 41–53. xiv, 16, 30

[Kochenderfer *et al.*, 2008b] Kochenderfer, M.J., Espindle, L.P., Kuchar, J.K. & Griffith, J.D. (2008b). Correlated encounter model for cooperative aircraft in the national airspace system version 1.0. Project Report ATC-344, MIT Lincoln Laboratory. 31, 37

[Kochenderfer *et al.*, 2008c] Kochenderfer, M.J., Kuchar, J.K., Espindle, L.P. & Gertz, J.L. (2008c). Preliminary uncorrelated encounter model of the national airspace system. Project Report CASSATT-1, MIT Lincoln Laboratory. 35

[Kochenderfer *et al.*, 2008d] Kochenderfer, M.J., Kuchar, J.K., Espindle, L.P. & Griffith, J.D. (2008d). Uncorrelated encounter model of the national airspace system version 1.0. Project Report ATC-345, MIT Lincoln Laboratory. 31

[Kochenderfer *et al.*, 2010a] Kochenderfer, M.J., Chryssanthacopoulos, J., Kaelbling, L. & Lozano-Perez, T. (2010a). Model-based optimization of airborne collision avoidance logic. Project Report ATC-360, MIT Lincoln Laboratory. xv, 16, 29, 72, 74, 80

[Kochenderfer *et al.*, 2010b] Kochenderfer, M.J., M. Edwards, M.W., Espindle, L.P., Kuchar, J.K. & Griffith, J.D. (2010b). Airspace encounter models for estimating collision risk. *Journal of Guidance, Control, and Dynamics*, **33**, 487–499. 16

[Kochenderfer *et al.*, 2011] Kochenderfer, M.J., Chryssanthacopoulos, J.P. & Weibel, R.E. (2011). A new approach for designing safer collision avoidance systems. *Air Traffic Control Quarterly*, **20**, 27–45. xiv, 25, 138

[Kochenderfer *et al.*, 2012] Kochenderfer, M.J., Holland, J.E. & Chryssanthacopoulos, J.P. (2012). Next-generation airborne collision avoidance system. *Lincoln Laboratory Journal*, **19**, 17–33. 24, 143

[Kopřiva *et al.*, 2012] Kopřiva, Š., Šišlák, D. & Pěchouček, M. (2012). Sense and avoid concepts: Vehicle-based SAA systems (vehicle-to-vehicle). In *Sense and Avoid in UAS: Research and Applications*, 143–173, Wiley, Hoboken, NJ. 22

[Kuchar, 2005] Kuchar, J.K. (2005). Safety analysis methodology for unmanned aerial vehicle (UAV) collision avoidance systems. In *USA/Europe Air Traffic Management R&D Seminars*, Baltimore, MD. 15

[Kuchar & Yang, 2000] Kuchar, J.K. & Yang, L.C. (2000). A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems*, **1**, 179–189. 21, 22, 92

[Lai & Whidborne, 2010] Lai, C.K. & Whidborne, J. (2010). Aircraft route re-planning for a pop-up obstacle using a direct method. In *Control 2010, UKACC International Conference on*, Conventry. 111

[Lai & Whidborne, 2012] Lai, C.K. & Whidborne, J.F. (2012). Automated return-to-route maneuvers for unmanned aircraft systems. In *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, Williamsburg, VA. 88

[Lai *et al.*, 2012] Lai, J., Ford, J.J., Mejias, L., O'Shea, P. & Walker, R. (2012). See and avoid using onboard computer vision. In P. Angelov, ed., *Sense and Avoid in UAS*, 265–294, John Wiley & Sons, Ltd. 23

[LaValle, 2006] LaValle, S.M. (2006). *Planning algorithms*. Cambridge University Press. 26

[Luongo *et al.*, 2009] Luongo, S., Carbone, C., Corraro, F. & Ciniglio, U. (2009). An optimal 3D analytical solution for collision avoidance between aircraft. In *Aerospace conference, 2009 IEEE*, 1–9. xv, 60, 63, 64, 65

[Luongo *et al.*, 2011] Luongo, S., Di Vito, V., Fasano, G., Accardo, D., Forlenza, L. & Moccia, A. (2011). Automatic collision avoidance system: Design, development and flight tests. In *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th*, Seattle, WA. 144

[Menon *et al.*, 1999] Menon, P.K., Sweriduk, G.D. & Sridhar, B. (1999). Optimal strategies for free-flight air traffic conflict resolution. *Journal of Guidance, Control, and Dynamics*, **22**, 202–211. xiv, 42

[MIDCAS, 2012] MIDCAS (2012). Design - performance working paper. Working paper, MIDCAS. 143

[MIDCAS, 2013] MIDCAS (2013). Integration aspects—Sense. MIDCAS Workshop #4. 2

[Milam, 2003] Milam, M.B. (2003). *Real-Time Optimal Trajectory Generation for Constrained Dynamical Systems*. Ph.D. thesis, California Institute of Technology. 26

[Möckli, 2006] Möckli, M.R. (2006). *Guidance and Control for Aerobatic Maneuvers of an Unmanned Airplane*. Ph.D. thesis, Swiss Federal Institute Of Technology Zurich. 46

[Murray, 2010] Murray, R.M. (2010). *Optimization-Based Control*. California Institute of Technology. 98, 125

[NATO, 2008] NATO (2008). Sense and avoid requirements for unmanned aerial vehicle systems operating in non-segregated airspace. Document PFP(NNAG-JCGUAV)D(2008)0002, NATO. 2

[Nicoullaud, 2012] Nicoullaud, V. (2012). Working paper - collision avoidance concept and operational aspects considered in MIDCAS. Working Paper MIDCAS-T-0173, MIDCAS. 3, 4

[Nuic, 2012] Nuic, A. (2012). User manual for the base of aircraft data (BADA) revision 3.10. EEC Technical/Scientific Report 12/04/10-45, EUROCONTROL. 7, 41, 43, 44, 68, 162

[Patchett & Ansell, 2010] Patchett, C. & Ansell, D. (2010). The development of an advanced autonomous integrated mission system for uninhabited air systems to meet UK airspace requirements. In *Intelligent Systems, Modelling and Simulation (ISMS), 2010 International Conference on*, Liverpool. xiv, 4, 15, 18, 54

[Patel & Goulart, 2010] Patel, R.B. & Goulart, P.J. (2010). The design of trigger mechanisms for aircraft collision avoidance maneuvers. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Toronto, Ontario Canada. 22, 145, 150

[Patel *et al.*, 2009] Patel, R.B., Goulart, P.J. & Serghides, V. (2009). Real-time trajectory generation for aircraft avoidance maneuvers. In *AIAA Guidance, Navigaion, and Coontrol Conference*, Chicago, Illinois. 27

[Pellebergs, 2010] Pellebergs, J. (2010). The MIDCAS project. In *27th International Congress of the Aeronautical Sciences*, Nice, France. 15

[Petri & Spriesterbach, 2012] Petri, M. & Spriesterbach, T. (2012). ACAS X_U—Ensuring collision avoidance interoperability. Working Paper WP-ASP13-19, FAA. 2, 3, 15

[Rao, 2009] Rao, A.V. (2009). A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, **135**, 497–528. 27

[Raynaud & Arino, 2006] Raynaud, B. & Arino, T. (2006). Final report on the safety of ACAS II in the European RVSM environment. Final Report ASARP/WP9/72/D, EUROCONTROL. xiii, 15, 16, 72

[Ross & Fahroo, 2006] Ross, I.M. & Fahroo, F. (2006). Issues in the real-time computation of optimal control. *Mathematical and Computer Modelling*, **43**, 1172–1188. 26

[RTCA, 2013] RTCA (2013). Terms of reference rtca special committee 228 minimum performance standards for unmanned aircraft systems. Tech. rep. 15

[Saunders & Beard, 2008] Saunders, J. & Beard, R. (2008). Reactive vision based obstacle avoidance with camera field of view constraints. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii. 23

[Sellem-Delmar, 2010] Sellem-Delmar, S. (2010). Stakeholder requirement synthesis. Technical Note MIDCAS-T-0039, MIDCAS. 2, 16, 87, 88

[Shakernia *et al.*, 2007] Shakernia, O., Chen, W.Z., Graham, S., Zvanya, J., White, A., Weingarten, N. & Raska, V.M. (2007). Sense and avoid (SAA) flight test and lessons learned. In *AIAA Infotech@ Aerospace Conference and Exhibit*, 1–12, Rohnert Park, California. 15

[Shin *et al.*, 2012] Shin, H.S., Tsourdos, A. & White, B. (2012). UAS conflict detection and resolution using differential geometry concepts. In P. Angelov, ed., *Sense and Avoid in UAS*, 175–204, Wiley. 24

[Singla & Singh, 2008] Singla, P. & Singh, T. (2008). A novel coordinate transformation for obstacle avoidance and optimal trajectory planning. In *AIAA Astrodynamics Specialist Conference and Exhibit*, Honolulu, Hawaii. 27

[Snell *et al.*, 1992] Snell, S.A., NNS, D.F. & ARRARD, W.L. (1992). Nonlinear inversion flight control for a supermaneuverable aircraft. *Journal of Guidance, Control, and Dynamics*, **15**, 976–984. 46

[Stevens & Lewis, 2003] Stevens, B.L. & Lewis, F.L. (2003). *Aircraft Control and Simulation*. Wiley, 2nd edn. 41, 43, 126

[Sundqvist, 2005] Sundqvist, B.G. (2005). Auto-acas - robust nuisance-free collision avoidance. In *44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05*, 3961–3963, Seville, Spain. 2

[Tadema, 2011] Tadema, J. (2011). *Unmanned Aircraft Systems HMI and Automation - Tackling Control, Integrity and Integration Issues*. Ph.D. thesis, Delft University of Technology. 20

[Temizer, 2011] Temizer, S. (2011). *Planning Under Uncertainty for Dynamic Collision Avoidance*. Ph.D. thesis, Massachusetts Institute of Technology. 4, 29

[Temizer *et al.*, 2010] Temizer, S., Kochenderfer, M.J., Kaelbling, L.P., Lozano-Pérez, T. & Kuchar, J.K. (2010). Collision avoidance for unmanned aircraft using Markov decision processes. In *AIAA Guidance, Navigation, and Control Conference*, Toronto, Ontario Canada. 55

[Turner *et al.*, 2012] Turner, R., Lehmann, R., Wadley, J., Kidd, D., Swihart, D., Bier, J. & Hobbs, K. (2012). Automatic aircraft collision avoidance algorithm design for fighter aircraft. In *Asia-Pacific International Symposium on Aerospace Technology*, DTIC Document, Jeju, Korea. 2

[Walters, 2012] Walters, E. (2012). Concept of operations for the Airborne Collision Avoidance System X. Concept of operations document, FAA. 2

[Weibel *et al.*, 2011] Weibel, R.E., Edwards, M. & Fernandes, C. (2011). Establishing a risk-based separation standard for unmanned aircraft self separation. In *Ninth USA/Europe Air Traffic Management Research & Development Seminar*, Berlin, Germany. 12, 13, 20

[Wolf & Kochenderfer, 2011] Wolf, T.B. & Kochenderfer, M.J. (2011). Aircraft collision avoidance using Monte Carlo real-time belief space search. *Journal of Intelligent & Robotic Systems*, **64**, 277–298. 4

[Yakimenko, 2000] Yakimenko, O.A. (2000). Direct method for rapid prototyping of near-optimal aircraft trajectories. *Journal of Guidance, Control, and Dynamics*, **23**, 865–875. 27, 97, 111

[Zeitlin, 2012] Zeitlin, A. (2012). Performance tradeoffs and the development of standards. In *Sense and Avoid in UAS: Research and Applications*, 35–54, Wiley. 2, 13, 15, 17, 56, 57

[Zeitlin *et al.*, 2006] Zeitlin, A.D., Lacher, A., Kuchar, J.K. & Drumm, A.C. (2006). Collision avoidance for unmanned aircraft : Proving the safety case. Tech. Rep. MP060219 (MITRE)/42PM ATC-329 (LL), The MITRE Corporation/MIT Lincoln Laboratory. 15