Spring 5-13-2016

Simulation and Performance Evaluation of Algorithms for Unmanned Aircraft Conflict Detection and Resolution

Jeffrey H. Ledet
University of New Orleans, jhledet@uno.edu

Simulation and Performance Evaluation of Algorithms for Unmanned Aircraft Conflict Detection and Resolution

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Engineering
Electrical

by

Jeffrey Harrer Ledet

B.S. University of New Orleans, 2014

May, 2016

Acknowledgement

I would like to thank my advisors, Dr. Jilkov and Dr. Li, for their assistance and mentorship throughout my Master's program. Dr. Jilkov was pivotal in shaping my mind to become what it is today. I have been in several of his classes and also worked with him as an RA for three years now, and I can honestly say that I deeply enjoyed being in his presence every step of the way. I would also like to thank Dr. Azzam, Dr. Charalampidis, Dr. Alsamman, Dr. Jovanovich, and Dr. Chen of UNO's Department of Electrical Engineering for their guidance and support throughout my time in the department. Lastly, I have to give a huge thanks to my family especially my parents for their tremendous love and support. Without them, I would not have been able to get this far.

Contents

List of Figures

Abstract

The problem of aircraft conflict detection and resolution (CDR) in uncertainty is addressed in this thesis. The main goal in CDR is to provide safety for the aircraft while minimizing their fuel consumption and flight delays. In reality, a high degree of uncertainty can exist in certain aircraft-aircraft encounters especially in cases where aircraft do not have the capabilities to communicate with each other. Through the use of a probabilistic approach and a multiple model (MM) trajectory information processing framework, this uncertainty can be effectively handled. For conflict detection, a randomized Monte Carlo (MC) algorithm is used to accurately detect conflicts, and, if a conflict is detected, a conflict resolution algorithm is run that utilizes a sequential list Viterbi algorithm. This thesis presents the MM CDR method and a comprehensive MC simulation and performance evaluation study that demonstrates its capabilities and efficiency.

Keywords: conflict detection and resolution, collision avoidance, probability of conflict, Viterbi algorithm, model predictive control, sense-and-avoid, Monte Carlo, multiple model

Chapter 1

Introduction

1.1 Aircraft Conflict Detection and Resolution (CDR) Problem

As time progresses, global airspace is expected to become more densely packed. To address this issue, in 2012, the United States government began the implementation of a new National Airspace System called the Next Generation Air Transportation System (NextGen) [1]. The objectives of NextGen include reducing fuel consumption, reducing time delays for departures and arrivals, and increasing the density of the airspace without compromising essential safety standards. Aircraft conflict detection and resolution (CDR) is one of the underlying fields of research that will help further the development of a system such as NextGen.

The problem considered in this thesis is the unmanned aerial vehicle (UAV) sense-and-avoid (SA) problem. In an SA scenario, a UAV, referred to as the own-ship and denoted by A, is flying around in some airspace that may also contain civilian (private or commercial) or military aircraft, referred to as intruders and denoted by $B^{(b)}$, $b = 1, 2, \ldots$. While the own-ship is strictly considered to be a UAV, intruders can be human operated aircraft or other UAVs. Within this scenario, the own-ship and the intruders are not capable of communicating with each other, and, therefore, no flight intent information can be shared between them.

Because of this lack of communication and the knowledge that some intruders may have human lives onboard, the own-ship has a major responsibility to ensure that it does not collide with any of the intruders or even come close to colliding for that matter. It realizes this collision avoidance (CA) responsibility by using its onboard sensors to detect the current state of intruders that are in its nearby region, propagating their trajectories into the future, evaluating the probability of a conflict occurring within the foreseen future, and executing any maneuvers that may be necessary to maintain strict safety requirements all while trying to get to its next waypoint. The Federal Aviation Administration (FAA) sets the minimum safety separation requirements that all aircraft must adhere to which are currently five miles of horizontal separation and 2,000 feet of vertical separation [2].

Real-world sensors are prone to measurement error, therefore the current states of the intruders that the own-ship receives are only estimates of their true state. Because the intruders' flight intents are unknown to the own-ship, their future states (i.e., possible future trajectories) are probabilistically distributed based on the current state estimates. Using these probability distributions, a probability of conflict (PC) can be evaluated and used to determine if an avoidance maneuver is necessary. If the PC exceeds some specified safety threshold, then the own-ship must compute an avoidance maneuver that resolves the predicted conflict.

An avoidance maneuver can consist of maneuvering strictly in the horizontal plane, strictly in the vertical plane, a combination of both horizontal and vertical motion, or even changing the aircraft's speed. In this work, only strictly horizontal and strictly vertical maneuvers were considered.

A major challenge of air traffic management (ATM) is to achieve autonomous CDR on each individual aircraft at short and mid-ranges without the intervention of air traffic controllers. As defined in the 2011 NextGen Avionics Roadmap [29], "In self-separation airspace, capable aircraft, equipped with Automatic Dependent Surveillance-Broadcast (ADS-B) and onboard conflict detection and alerting, are responsible for separating themselves from one another".

1.2 Model Predictive Control (MPC)

Model predictive control (MPC) is one method that is used to solve the CDR problem. In MPC, a dynamic model is used to predict the future events of a system in order to optimize a control action. At every time step k, an optimal control strategy (i.e., sequence of control actions) is computed for the duration of a finite time horizon, k to $k+N$. Upon acquiring a control strategy, only the first action within the strategy is executed. The time horizon is then shifted forward one time step, and the process starts all over again. Because the time horizon is iteratively shifted forward in time, this method is also referred to as receding horizon control. While MPC does not generally produce optimal solutions, it is very useful for online applications in which a fast response is required.

CDR is typically performed at three different scales of the look-ahead time horizon [30]. The first scale is long range where CDR is carried out over the time horizon of several hours across the entire national airspace. This scale is used for the coordination of daily flight plans and schedules to ensure that no conflicts exist at airports or en route sectors. The second scale is mid-range where CDR is carried out over the time horizon of tens of minutes across smaller, more localized regions. This scale is used to update flight plans mid-flight via communication with an air traffic controller (ATC) to maintain the required safety separations. The third scale is short range where CDR is carried out over the time horizon of seconds to minutes. This scale is used to mitigate unforeseen, near-future conflicts via an onboard collision avoidance system (CAS) on the fly.

1.3 Overview of Existing Methods in the Literature

In the literature, many different approaches exist that attempt to solve the CDR problem. The most common categories of CDR methods are geometric, force field, airspace discretization, mixed-integer linear programming (MILP), and probabilistic [6, 12, 19].

Among all of the categories of CDR methods listed, geometric methods are the simplest and most straightforward theoretically. Using linear projections, the future trajectories of any aircraft in the encounter are predicted. One geometric method is called the point of closest approach (PCA) method [28], and it consists of comparing the velocity vectors of the two aircraft to find the PCA which is then used to find the miss distance vector between them. If the length of the miss distance vector falls below the minimum separation distance, then the two aircraft will be turned away from each other in order to increase the space between them.

In the PCA method, some coordination is required because one aircraft will be turned in one direction and the other in the opposite direction. Furthermore, this method is only optimal for encounters involving only two aircraft. As the number of aircraft in an encounter increases beyond two, the method becomes gradually worse because the act of two aircraft fixing a predicted conflict between each other may introduce a new conflict between a different pair of aircraft.

One geometric method that does not require any coordination is called the collision cone method [25]. In this method, a circle is placed around the obstacle or aircraft that is to be avoided. The collision cone is formed by two lines that go from the tangents of the circle to the aircraft that is doing the avoiding. If the velocity vector of the aircraft that is doing the avoiding is between the two tangential lines, then a future conflict has been detected, and a resolution needs to be made. In this method, a combination of velocity, heading, and altitude changes can be used to mitigate the conflict. However, the simplest resolution is to make the velocity vector of the aircraft doing the avoiding match one of the two tangential lines.

The collision cone method is typically used to avoid static objects but can be extended for avoidance of dynamic objects. Despite this method not requiring any coordination, it still suffers as the number of aircraft in an encounter increases beyond two in the same way that the PCA method does. One way in which the PCA method and collision cone method differ is that the PCA method produces only one option to resolve the conflict (i.e., turn in a particle direction), whereas the collision cone method produces multiple options, some better than others, to resolve the conflict.

Another popular geometric method which is quite similar to PCA is the reactive inverse proportional navigation method (RIPNA) [10,13]. RIPNA is derived from the Proportional Navigation (PN) guidance law which is used to make missiles collide with their targets. The PN guidance law applies a lateral acceleration

to a missile in order to make the Line of Sight (LOS) rate of rotation between the missile and the target zero. Making the LOS rate of rotation zero effectively means that the missile is heading directly towards its target, and a collision is imminent.

RIPNA is essentially an inverse PN guidance law meaning that lateral accelerations are applied that will increase the LOS rate of rotation between two aircraft thus pulling them out of a collision course. For each pair of aircraft in an encounter, a Zero Effort Miss (ZEM) distance, the minimum distance that two aircraft will reach if they both remain on their current trajectories, is computed along with a time-to-go, the time until the ZEM is reached. The currently selected aircraft will only consider other aircraft with which it has a ZEM less than some specified desired distance as potential threats. From these potential threats, the aircraft that it has the smallest time-to-go with is avoided first followed by the aircraft with the second smallest time-to-go and so on.

Just like with PCA, RIPNA poses the issue of resolving one conflict between a pair of aircraft only to create a new conflict. Furthermore, while RIPNA can maintain the minimum separation distance between aircraft, it cannot guide an aircraft to its next waypoint, and, in [10, 13], a separate guidance method called Dubins path is used to guide an aircraft to its destination when no conflicts exist. Dubins path states that an aircraft must turn at its maximum turn rate until its heading faces its destination. A special case exists where an aircraft's destination lies within the circle defined by the aircraft's maximum turn rate. To resolve this case, the aircraft must turn in the opposite direction until the destination lies on the edge or outside of the maximum turn radius circle.

The most novel of the CDR methods fall arguably into the force field category. This CDR category is derived from particle physics where repulsive and attractive forces dictate the interaction between charged particles. Aircraft, obstacles, and aircraft destinations all act as charged particles. Aircraft and obstacles are assigned repulsive charges, and the aircraft destinations are assigned positive charges. When an aircraft feels the charge of another aircraft, an obstacle, or its destination, a force vector is computed using the force vector function.

A force field method's performance is solely dependent on the force vector function that it uses. This function must be continuous and differentiable, must produce larger force vectors as the distance between an aircraft and an obstacle decreases, and must produce smaller force vectors as the distance between an aircraft and its destination decreases. Generally speaking, a force vector function is difficult to design and implement [34, 37]. However, after obtaining a sufficient force vector function, force field methods benefit from fast calculations. Such methods can reliably produce safe paths for aircraft to fly when the number of aircraft is relatively small, although the safe paths are often not optimal. Because force field methods are purely reactive, the safe paths of aircraft become drastically less optimal as the number of aircraft increases.

A few special cases exist within the force field methods. One special case occurs when an aircraft gets stuck in a local minimum which is where the net force is either zero or very close to zero. At the expense of computation time, when an aircraft is found to be stuck in a local minimum, some additional algorithm is run to help the aircraft escape the local minimum. Another special case arises when generated waypoints are unreachable or extremely difficult to reach based on the aircraft's performance specifications, specifically maximum turning radius.

Another category of CDR methods revolves around airspace discretization. By dividing the airspace into discrete, uniformly spaced nodes, the CDR problem becomes one of finding an optimal path through a weighted graph. Popular computer science search algorithms such as Dijkstra's algorithm and A* search can be used to efficiently find the shortest route between two points in a graph.

Dijkstra's algorithm finds the shortest path from a given source node to every other node in a graph [8]. When only a single shortest path is being sought, i.e., a single destination node is specified, the algorithm stops searching when the shortest path from the source node to the destination has been found. Dijkstra's algorithm does not directly search in the direction of the destination node but instead gradually explores, in a circular wavefront-like fashion, farther and farther away neighboring nodes and computing the shortest path to each consecutive neighboring node.

A* search, an extension of Dijkstra's algorithm, uses heuristics to make the search more efficient. Within the CDR framework, one heuristic would be a constraint on the flying direction [35]. This constraint would limit the searched neighboring nodes to only those that fall within a $\pm 90^\circ$ radius of the direction of the current velocity vector thus reducing the total number of nodes searched by one-third. The performance of A* search is entirely dependent on the heuristics that are used. While Dijkstra's algorithm guarantees that the solution is strictly optimal, A* search does not always guarantee the same optimality but benefits from a reduction in computational complexity.

A derivation of A* search called Aϵ^* works by only exploring neighboring nodes that are within a fixed, positive ϵ value of the lowest-cost neighboring node [7]. By searching fewer nodes than A*, Aϵ^* further reduces the computational complexity at the expense of optimality. Fig. 1.1 shows a comparison of the paths generated by Dijkstra's algorithm, A*, and Aϵ^*.

In [23], the CDR problem is formulated as a classic stochastic optimal control problem where an optimal control input is computed by approximating the stochastic differential equation of motion by a Markov chain. The method involves time and state-space discretization which makes the computational complexity quite high. The major caveat with methods that discretize the state-space is the tradeoff between the resolution of the discretized grid, i.e., the amount of nodes per unit length of the grid, and the computational complexity. In a two-dimensional setting, N^2 amount of nodes will exist within the grid.

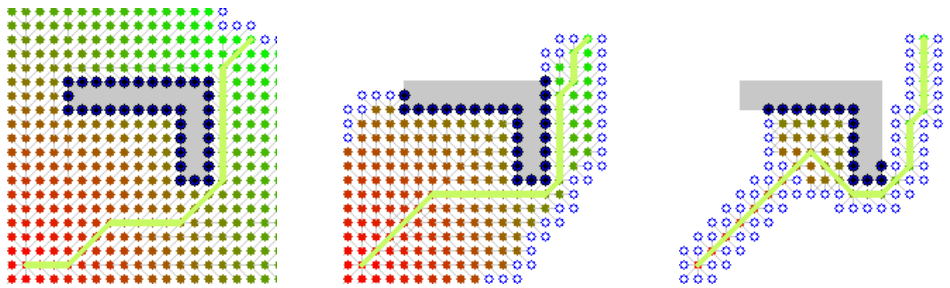

Figure 1.1: Paths generated by Dijkstra's algorithm (left), A* search (middle), and Aϵ^* search (right).

As the number of nodes per unit length increases, the computational burden of finding an optimal solution increases exponentially. On the other hand, if the state-space is divided up into fewer but larger nodes, then the optimality and safety of the generated path will be compromised.

The next category of CDR methods utilize the powerful optimization technique that is MILP [7, 12, 32]. MILP models obstacles, collision avoidance rules, and no-fly zones as logical (integer) constraints and an aircraft's performance limitations as continuous constraints. Open-source and commercial optimization software packages are available that can solve the full set of equations and produce minimum flight-time, collision-free trajectories for all aircraft involved in the scenario. In the MILP framework, the trajectories of all aircraft in an encounter are optimized jointly, in other words for the "greater good."A trajectory optimized in this way may be suboptimal for an individual aircraft, but the sum of the trajectories of all aircraft is globally optimal.

The attractive feature of MILP is that it is guaranteed to find the most optimal feasible solution granted that at least one feasible solution exists. The significant drawback of MILP methods is that they are practically infeasible for real-time CA because the computation time is too high. On top of that, the time complexity to solve a set of MILP equations increases as more constraints are added. This increase in time complexity contrasts with a method such as A* search which would benefit from an increase in constraints.

The last CDR category that will be discussed consists of the probabilistic methods. Probabilistic methods use the uncertainties inherent in the dynamic model to predict the possible future trajectories of aircraft over a finite time horizon. Each possible future trajectory is weighted depending on its probability of occurring. Probabilistic approaches are reasonable because they provide a good tradeoff between relying too heavily on an aircraft sticking to its original flight plan vs. relying too heavily on an aircraft performing worst-case maneuvers. Within a probabilistic approach, decisions for CR are made based on the fundamental likelihood of a conflict, i.e., the PC.

Some probabilistic methods model the problem as a discrete-state partially observable Markov deci-

sion process (POMDP) [3,38]. In a Markov decision process (MDP), the state of the system (i.e., positions and velocities of all aircraft) changes probabilistically based on the current states and actions of the aircraft involved. In a POMDP, the state of the system is now expressed in terms of observations (i.e., an estimation of an aircraft's state) which are probabilistically generated and conditioned on the current states and actions of the aircraft involved. The observations all together form a belief, a probability distribution over all possible system states. Solving a POMDP consists of computing a policy that selects actions in a way that either maximizes an expected reward or minimizes an expected cost. The policy accounts for the current uncertainty in the system (i.e., exact positions and velocities of intruder aircraft are not exactly known) and the future uncertainty about how the system will evolve (i.e., what maneuvers might the intruders make).

POMDP algorithms generally require discretization of the state-space in order to be able to compute an optimal control policy. State-space discretization imposes a computational burden that can render these types of algorithms useless for practical implementation. By using Monte Carlo (MC) methods, the computational problems introduced by discretizing the state-space can be relieved. In [3], a MC Value Iteration algorithm was used to solve the POMDP in the continuous state-space.

MC methods account for probabilistic uncertainty due to disturbances, uncertain state estimation, modeling error, and stochastic mode transitions. As a result, they are well suited for solving chance-constrained (i.e., the probability of failure must be below a certain threshold) stochastic optimal control problems. A finite number of particles are used to approximate all aircraft states as probability distributions. Through this approximation, an intractable stochastic optimization problem can be transformed into a tractable deterministic optimization problem. By solving the deterministic problem, a solution that approximates the original stochastic problem can be found. The approximation error decreases as the number of particles increases. Also, particle-based methods such as MC are able to handle arbitrary probability distributions.

[4,9,18] proposed solutions to the stochastic MPC problem by using sequential Monte Carlo (SMC). SMC is similar to simulated annealing in that it is capable of producing a globally optimum solution given a non-convex optimization problem that may have several local minima or maxima. SMC extends MC by adding a resampling process wherein the particles of a probability distribution are continually resampled to allow them to converge to the global optimizer. Despite MC methods fixing the problem of having to discretize the state-space, their computational complexity can still be very high. The body of work within this thesis follows a probabilistic approach based around MC which provides a general and systematic way to handle the uncertainties that exist within the CDR problem.

Fig. 1.2 shows the process of CDR for a probabilistic method. The first step for the own-ship is to estimate the current states of all aircraft in the surrounding environment. Using the current estimates and

a dynamic model, the own-ship can find the predicted states of the detected aircraft. For conflict detection, the predicted states are used to obtain a PC between the own-ship and every other aircraft. If any one of the calculated PCs exceeds a specified safety threshold, then conflict resolution is required. Otherwise, the own-ship and all other aircraft progress one time step into the future along their original flight paths.

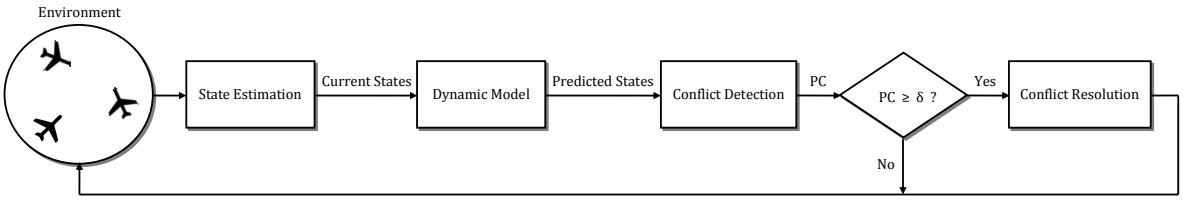

Figure 1.2: Conflict Detection and Resolution Flowchart.

Across all of the categories of CDR methods, many ways exist for projecting an intruder's current information into the future. The four main methods of trajectory propagation are nominal, worst-case, probabilistic, and flight-plan sharing. The nominal method projects current states into the future without the consideration of disturbances or flight-plan changes. An intruder's position is simply extrapolated based on it's current velocity vector or rate of turn. The worst-case method assumes that an intruder will perform any range of maneuvers with equal probability. This method is extremely conservative because in some cases a conflict may be declared for a projected trajectory that is very unlikely to happen in a real-world situation. The probabilistic method constructs a set of possible future trajectories where each trajectory is assigned a probability of occurring. In the flight-plan sharing method, all aircraft share their flight intents with one another. This sharing of information makes the CA problem mush easier. As technology progresses and becomes more widely available, the flight-plan sharing method will see more and more attention. Fig. 1.3 illustrates the four main trajectory propagation methods.

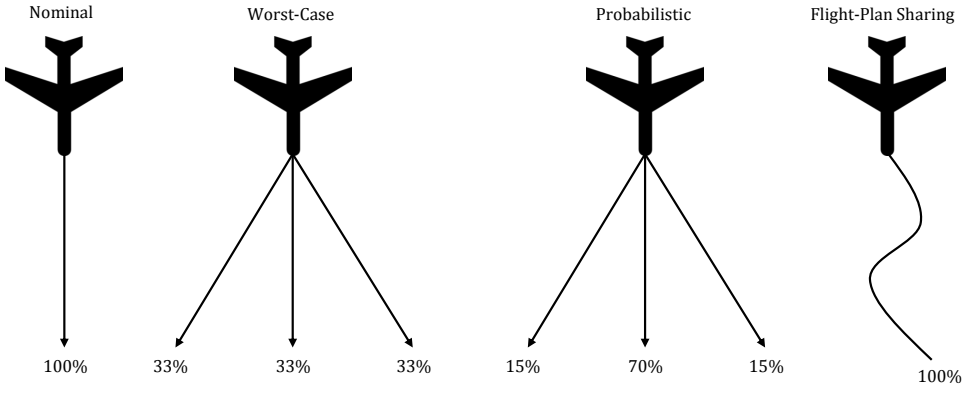

Figure 1.3: Four main methods for trajectory propagation.

Chapter 2

Multiple Model (MM) CDR Framework

The MM framework and algorithms for CDR presented in this thesis were originally proposed and developed in our papers [15–17].

2.1 Aircraft Model

The characteristic motion models of an aircraft can be classified into the following kinematic categories [20]. In the horizontal plane, the categories are constant velocity (CV) (i.e., straight line motion), constant acceleration (CA) (i.e., speed up/slow down), and constant turn (CT) (i.e., left/right). In the vertical plane, the categories are constant height (CH) (i.e., level cruise) and constant climb/descent (CD). A flight mode is then modeled as a combination of a horizontal and a vertical model. Flight dynamics are described by a hybrid system (HS) model which includes a set of flight models and the rules that govern the transition between these flight models.

Consider the Markov Jump Linear System (MJLS) shown below:

$$x_k = F_{m_k} x_{k-1} + G_{m_k} w_k \tag{2.1}$$

$$z_k = H_{m_k} x_k + v_k \tag{2.2}$$

where $k = 1, 2, \ldots$ is the time index, $x_k \in \mathbb{R}^{n_x}$ is the continuous kinematic state, $m_k \in \mathbb{M} = \left\{m^{(1)}, m^{(2)}, \ldots, m^{(M)}\right\}$ is the discrete modal state, $z_k \in \mathbb{R}^{n_z}$ is the sensor measurement, and $w_k \sim \mathcal{N}(0, Q_k)$ and $v_k \sim \mathcal{N}(0, R_k)$ are mutually independent white process and measurement noises, respectively. For the own-ship, the model sequence $\langle m_k \rangle$ is determined by the own-ship itself. While for any intruder, the model sequence $\langle m_k \rangle$ is

assumed to be a Markov chain with the following transition and initial probabilities:

$$P\{m_k = m^{(j)} | m_{k-1} = m^{(i)}\} = \pi_{ij} \tag{2.3}$$

$$P\{m_0 = m^{(i)}\} = \mu_0^{(i)} \tag{2.4}$$

A CA kinematic model is not used in this work. However, CV (2.5) and CT (2.6) kinematic models are used and are shown below [20]:

$$x_k = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix} w_k \tag{2.5}$$

$$x_k = \begin{bmatrix} 1 & \frac{\sin \omega T}{\omega} & 0 & -\frac{1-\cos \omega T}{\omega} \\ 0 & \cos \omega T & 0 & -\sin \omega T \\ 0 & \frac{1-\cos \omega T}{\omega} & 1 & \frac{\sin \omega T}{\omega} \\ 0 & \sin \omega T & 0 & \cos \omega T \end{bmatrix} x_{k-1} + G w_k \tag{2.6}$$

where $x_k = [\ \mathsf{x} \quad \dot{\mathsf{x}} \quad \mathsf{y} \quad \dot{\mathsf{y}}\ ]'$ is the state vector, T is the time step duration, ω is a constant turn rate (i.e., control input), and w_k is the process noise. Note that CV $= \lim_{\omega \to 0} \mathrm{CT}(\omega)$.

The CH/CD model that is used is shown below:

$$x_k = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} v_k + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \end{bmatrix} w_k \tag{2.7}$$

where $x_k = [\ \mathsf{x} \quad \dot{\mathsf{x}} \quad \mathsf{z}\ ]'$ is the state vector, T is the time step duration, $v_k \in \left\{v^{(1)}, \ldots, v^{(5)}\right\}$ are the constant climb/descent rates (i.e., control inputs), and w_k is the process noise.

2.2 Multiple Model (MM) Trajectory Prediction

Multiple model (MM) trajectory prediction is the new and innovative way to handle the many challenges that exist in aircraft trajectory prediction. It is widely accepted because of its ability to effectively handle multiple modes of operation [21]. It is also capable of providing good approximations of highly nonlinear and non-Gaussian state distributions that are a result of significant uncertainties in the dynamic model such

as aircraft intent and various flight modes. Fig. 2.1 illustrates an SA scenario where the own-ship needs to predict the trajectory of an intruder under uncertainty of the intruder's intent (for possible maneuvers) in order to avoid a conflict.

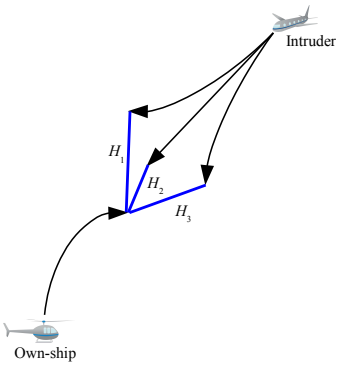

Figure 2.1: MM Trajectory Prediction in a "sense-and-avoid" Scenario. H_1, H_2, and H_3 denote possible hypotheses on the intruder's predicted trajectories.

In a probabilistic trajectory prediction, the main goal is to approximate as accurately as possible the PDFs $f(x_{k+n}|z^k)$, $n = 1, \ldots, N$, where k is the current time, $z^k = \{z_1, \ldots, z_k\}$ is the available sensor data, and N is the look-ahead time horizon for how far into the future predictions are made. MM filters such as GPB, IMM, or VSMM [21] provide state estimates $\hat{x}_{k|k}^{(i_k)}$, associated covariances $P_{k|k}^{(i_k)}$ and model probabilities $\mu_{k|k}^{(i_k)}$, $i_k = 1, \ldots M$ where M is the number of discrete modal states. The MM filter density is approximated as a Gaussian mixture (GM):

$$f(x_k|z^k) = \sum_{i_k=1}^{M} \mu_{k|k}^{(i_k)} \mathcal{N}\left(x_k; \hat{x}_{k|k}^{(i_k)}, P_{k|k}^{(i_k)}\right) \tag{2.8}$$

Furthermore, the prediction is based on the motion model given by (2.1) and (2.3) with $\hat{x}_{k|k}^{(i_k)}$, $P_{k|k}^{(i_k)}$, and $\mu_{k|k}^{(i_k)}$ being the "initial" condition.

Let a model sequence in the time interval $[k, k+N]$ be denoted by

$$\mathfrak{M}_k^{k+N} \triangleq (m_k^{(i_k)}, \ldots, m_{k+N}^{(i_{k+N})}) \in \mathbb{M}^{N+1} \tag{2.9}$$

where $i_{k:k+N} \triangleq (i_k, \ldots, i_{k+N})$ is the underlying sequence of model indices.

Then

$$f(x_{k+N}|z^k) = \sum_{i_{k:k+N}} f(x_{k+N}|\mathfrak{M}_k^{k+N}, z^k) P\{\mathfrak{M}_k^{k+N}|z^k\} \tag{2.10}$$

where

$$f(x_{k+N}|\mathfrak{M}_k^{k+N}, z^k) = \mathcal{N}\left(x_{k+N}; \hat{x}_{k+N|k}^{(i_{k:k+N})}, P_{k+N|k}^{(i_{k:k+N})}\right) \tag{2.11}$$

$$P\{\mathfrak{M}_k^{k+N}|z^k\} = \mu_{k|k}^{(i_k)} \pi_{i_k i_{k+1}} \dots \pi_{i_{k+N-1} i_{k+N}} \tag{2.12}$$

and $\hat{x}_{k+N|k}^{(i_{k:k+N})}, P_{k+N|k}^{(i_{k:k+N})}$ are obtained recursively by the Kalman filter prediction equations for each model sequence $\mathfrak{M}_k^{k+N}$ with the initial $\hat{x}_{k|k}^{(i_k)}, P_{k|k}^{(i_k)}$.

2.3 Weather Model

Along with aircraft-aircraft encounters, aircraft-weather encounters are also taken into account in which the own-ship is given data of bad weather cells that lie along its intended flight path. As in the aircraft-aircraft encounters, the own-ship must perform any necessary maneuvers to avoid conflicts with the bad weather. Based on four-dimensional predicted weather data (available from a weather service repository, such as the METOC Data Server (MDS)), the spatial density and concentration (severity) of the bad weather can be obtained. Modeling the bad weather concentration as a spatial random variable ξ, the spatial density can be viewed as a probability density, and the PDFs of ξ can be constructed by GM fitting of the spatial density data:

$$f_\xi(p_{k+n}) = \sum_{l=1}^{L_{k+n}} \mu_{k+n}^{(l)} \mathcal{N}(p_{k+n}; \bar{p}_{k+n}^{(l)}, C_{k+n}^{(l)}) \tag{2.13}$$

where $n = 1, \dots, N$ and $p = [\mathsf{x}\ \mathsf{y}\ \mathsf{z}]'$ is the position vector. Based on (2.13), aircraft-weather CDR can be done exactly as aircraft-aircraft CDR is done based on (2.10).

Chapter 3

Conflict Detection (CD)

The MM-based approach and algorithm for CD presented in this chapter were proposed in our paper [16].

3.1 Predicted Probability of Conflict (PC)

Most probabilistic methods for estimating the probability of conflict (PC) in the literature assume that the predicted separation vector between two aircraft is a Gaussian distribution. In an advanced multiple model trajectory prediction framework, however, the separation vector has a Gaussian mixture distribution, and approximating it by a single Gaussian, as in [14, 22, 40], leads to significant inaccuracies of the predicted PC in a highly uncertain environment. This chapter presents a more accurate method, proposed in [16], for estimating PC by utilizing the information from multiple model aircraft trajectory prediction.

PC is the probability that the distance between two aircraft falls below a specified minimum separation distance. As specified by the current aviation standards, the horizontal (2D) and vertical separation distances (1D) are commonly treated separately, [27,30], which amounts to a cylindrical protected zone. A 3D ellipsoidal protected zone, proposed in [5,22], where the horizontal and vertical separations are treated jointly is considered here. More specifically, let A and B denote two aircraft with position vectors $x_A = (\mathsf{x}_A, \mathsf{y}_A, \mathsf{z}_A)'$ and $x_B = (\mathsf{x}_B, \mathsf{y}_B, \mathsf{z}_B)'$ in an inertial (East-North-Up) Cartesian coordinate system $O\mathsf{xyz}$, and let the distance vector be $\rho_{AB} = x_A - x_B$. If λ_{xy} and λ_{z} are the given horizontal and vertical separation thresholds, respectively, the "weighted" distance vector is $\Lambda\rho_{AB}$, where $\Lambda = \mathrm{diag}\{1/\lambda_{\mathsf{xy}}, 1/\lambda_{\mathsf{xy}}, 1/\lambda_{\mathsf{z}}\}$, and the ellipsoidal protected zone is defined as

$$\mathcal{R}^\lambda = \left\{\rho \in \mathbb{R}^3 : \|\Lambda\rho\| \le 1\right\} \tag{3.1}$$

where $\|a\| = (a'a)^{\frac{1}{2}}$ is the Euclidean norm of a vector a.

If $f_{\rho_{AB_k}}(\rho)$ is the probability density function (PDF) of the distance vector ρ_{AB_k} at time k, then the instantaneous PC is

$$PC_k = P\left\{\|\Lambda\rho_{AB_k}\| \le 1\right\} = \int_{\mathcal{R}^\lambda} f_{\rho_{AB_k}}(\rho)d\rho \tag{3.2}$$

and the maximal PC over a time interval $(k, k+N]$ is

$$PC_{(k,k+N]} = \max_{0<n\leq N} PC_{k+n} \tag{3.3}$$

In the CDR problem, k is the current time and N is the length of the time horizon. The corresponding predicted PC is computed via (3.2) and (3.3), respectively, based on predicted densities $\hat{f}_{\rho_{AB_{k+n}}}(\rho|z^k) \approx f_{\rho_{AB_{k+n}}}(\rho)$, $n = 1, 2, \ldots, N$, where $z^k = \{z_1, \ldots, z_k\}$ is the available sensor data.

Let

$$x_A \sim f_{x_A}(x) = \sum_{i=1}^{M_A} \mu_A^{(i)} \mathcal{N}\left(x; \hat{x}_A^{(i)}, P_A^{(i)}\right) \tag{3.4}$$

$$x_B \sim f_{x_B}(x) = \sum_{j=1}^{M_B} \mu_B^{(j)} \mathcal{N}\left(x; \hat{x}_B^{(j)}, P_B^{(j)}\right) \tag{3.5}$$

If x_A and x_B are independent, then ρ_{AB} also has a GM distribution, given by

$$\rho_{AB} \sim f_{\rho_{AB}}(\rho) = \sum_{l=1}^{M_{AB}} \mu_{AB}^{(l)} \mathcal{N}\left(\rho; \hat{\rho}_{AB}^{(l)}, P_{AB}^{(l)}\right) \tag{3.6}$$

where $M_{AB} = M_A M_B$, $\mu_{AB}^{(l)} = \mu_A^{(i)} \mu_B^{(j)}$, $\hat{\rho}_{AB}^{(l)} = \hat{x}_A^{(i)} - \hat{x}_B^{(j)}$, $P_{AB}^{(l)} = P_A^{(i)} + P_B^{(j)}$ and the single index $l = 1, \ldots, M_{AB}$ is correspondent to the double index (i, j), $i = 1, \ldots, M_A$, $j = 1, \ldots, M_B$.

3.2 Computing the Predicted PC

The randomized algorithm, given in Algorithm 1, is used to estimate the instantaneous PC_k. This algorithm is based on the direct estimation of PC by the definition (3.2) with (3.6).

Algorithm 1 Randomized Algorithm for Estimating PC_{k+n}

1: Choose N_{MC}, number of MC runs
2: Set $\widehat{PC}_{k+n} = 0$
3: **for** $r = 1$ to N_{MC} **do**
4: Sample random index $l \sim \left\{\mu_{AB_{k+n}}^{(l)} : l = 1, \ldots, M_{AB_{k+n}}\right\}$
5: Sample random vector $\rho \sim \mathcal{N}\left(\rho; \hat{\rho}_{AB_{k+n}}^{(l)}, P_{AB_{k+n}}^{(l)}\right)$
6: **if** $\|\Lambda\rho\| \leq 1$ **then**
7: $\widehat{PC}_{k+n} = \widehat{PC}_{k+n} + 1$
8: $\widehat{PC}_{k+n} = \widehat{PC}_{k+n}/N_{MC}$

The good approximation properties (in a probabilistic sense) of the randomized estimation of PC are discussed in [30, 31, 39], wherein quantitative bounds on the approximation error can be found. In the next section, the accuracy of this PC estimation method is evaluated through Monte Carlo simulation.

3.3 PC Estimation Accuracy Evaluation

The performance of the proposed GM-based PC prediction method is compared, through simulation, with a traditional Gaussian-based approach over an SA UAV encounter scenario. The results demonstrate and quantify the improvement achieved by the proposed CD method.

Several SA encounter scenarios, as illustrated in Fig. 2.1, were simulated where the own-ship predicts the trajectory of an intruder under uncertainty of the intruder's intent in order to avoid a conflict. More and more research has been focused on SA recently since SA capability is required for UAVs to be able to operate in civil airspace [38].

For simplicity, the scenarios are contained within the horizontal plane $O\mathsf{xy}$. The own-ship, A, is assumed to have a constant velocity (CV) motion, while the intruder, B, can have either a CV or a constant turn (CT) motion. An intruder B is modeled through the hybrid system (2.1), (2.3), (2.4) and has three motion models which are $m^{(1)} = \text{CV (straight)}$ with $\omega^{(1)} = 0\,^{\text{o}}/s$, $m^{(2)} = \text{CT (left turn)}$ with $\omega^{(2)} = 1\,^{\text{o}}/s$, and $m^{(3)} = \text{CT (right turn)}$ with $\omega^{(3)} = -1\,^{\text{o}}/s$. The flight mode initial and transition probabilities are $\mu_0^{(1)} = 0.8,\ \mu_0^{(2)} = 0.1,\ \mu_0^{(3)} = 0.1$ and $\pi_{11} = 0.8,\ \pi_{12} = 0.1,\ \pi_{13} = 0.1,\ \pi_{21} = 0.9,\ \pi_{22} = 0.1,\ \pi_{23} = 0.0,\ \pi_{31} = 0.9,\ \pi_{32} = 0.0,\ \pi_{33} = 0.1$. The time step duration is $T = 20\,\text{sec}$, and the process noise covariance is $Q = q^2 I$ with $q = 5m/s^2$.

Four scenarios are presented each with different aircraft geometries that are generated according to the aforementioned models. In each scenario, the amount of Monte Carlo runs (i.e., the number of random trajectories generated) is $N_{MC} = 1000$, and the time horizon is 60 sec. PC is estimated as the ratio N_C/N_{MC}, the number of runs where a conflict occurred N_C over the total number of runs performed N_{MC}. The minimum horizontal separation distance that defines a conflict is $\lambda_{\mathsf{xy}} = 1km$.

The predicted PC is computed using the proposed algorithm, Algorithm 1, which is based on MM trajectory prediction (2.10)–(2.12) and is referred to as MM prediction in all of the results. For the purpose of comparison, the predicted PC is also computed through the algorithm of [14, 22, 40] where MM trajectory prediction is still used but the predicted GM is approximated by a single Gaussian matching the mean and covariance of the mixture. PC predicted in this way is referred to as combined prediction in all of the results. With this Gaussian approximation, integral (3.2) is computed in the simulation by the randomized PC estimation algorithm of [30].

Fig. 3.1 shows the scatter plots for the ground truth (top), the combined prediction (middle), and the MM prediction (bottom) in Scenario 1. In this scenario, the intruder follows a nearly CV motion which means that it does not take any turns and is only acted upon by a random process noise. The combined prediction better approximates the true Gaussian distribution, i.e., ground truth, over MM prediction which

accounts for possible maneuvers made by the intruder. For this scenario, the true PC is 0.2, combined prediction PC is 0.24, and MM prediction PC is 0.15. In this particular case, the combined prediction achieved a slightly more accurate predicted PC than the MM prediction did because the intruder's true position density is already Gaussian.

Fig. 3.2 shows the results for Scenario 2. In this scenario, the intruder follows a random sequence of nearly CV and CT motion models that are chosen according to the transition probabilities previously provided. In this particular case, MM prediction better approximates the true Gaussian distribution over combined prediction because the true distribution here is in fact a GM. The combined prediction approximates the true PC fairly poorly. The true PC is 0.25, combined prediction PC is 0.365, and MM prediction PC is 0.248. As expected, when more uncertainty is present in a scenario, combined prediction fails as opposed to MM prediction which very accurately predicts PC.

Scenarios 3 and 4 are shown in Figs. 3.3 and 3.4, respectively. As observed in Scenario 2 and all other scenarios that involve random maneuvers for the intruder, MM prediction more accurately estimates the true PC than combined prediction does.

Finally, Fig. 3.5 compares, over all four scenarios, the predicted PC as computed by the proposed method (MM Prediction) and an existing method (Combined Prediction) against the true PC. For the case in which the intruder makes no maneuvers, both PC prediction methods are comparable. However, for the cases that involve more serious uncertainty about the intruder's motion intent, the proposed PC prediction method is considerably more accurate.

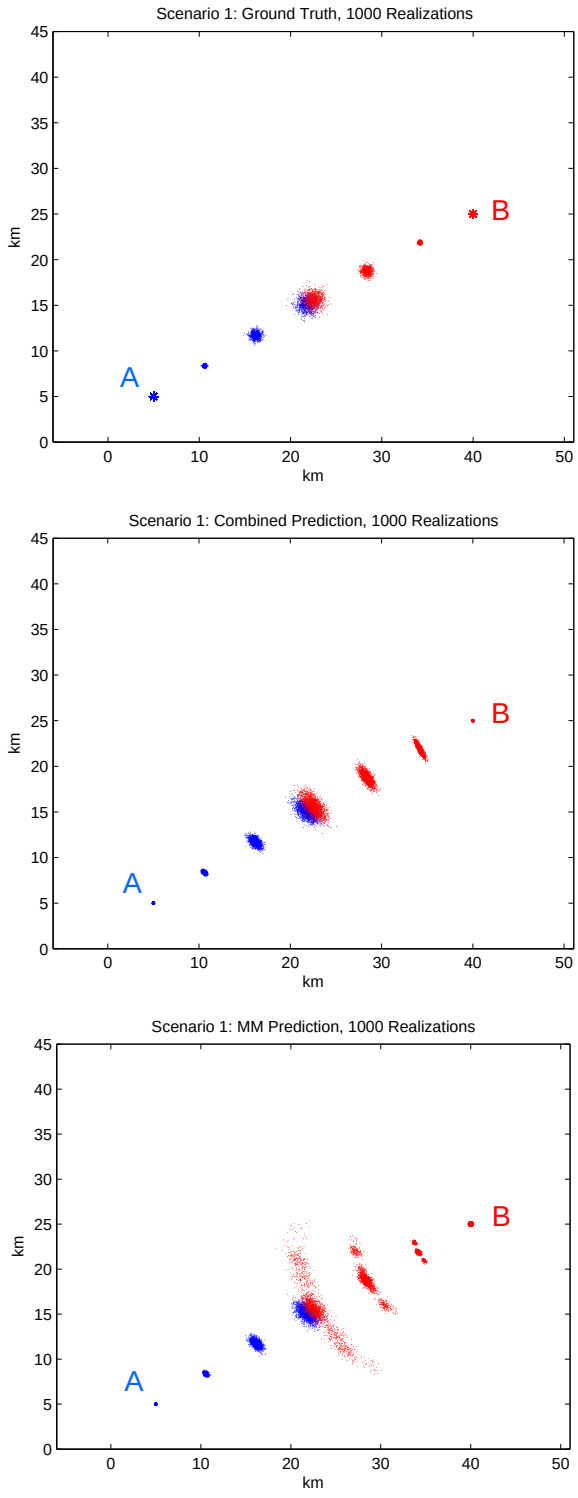

Figure 3.1: Scenario 1: Truth (top), Combined Prediction (middle) and MM Prediction (bottom)

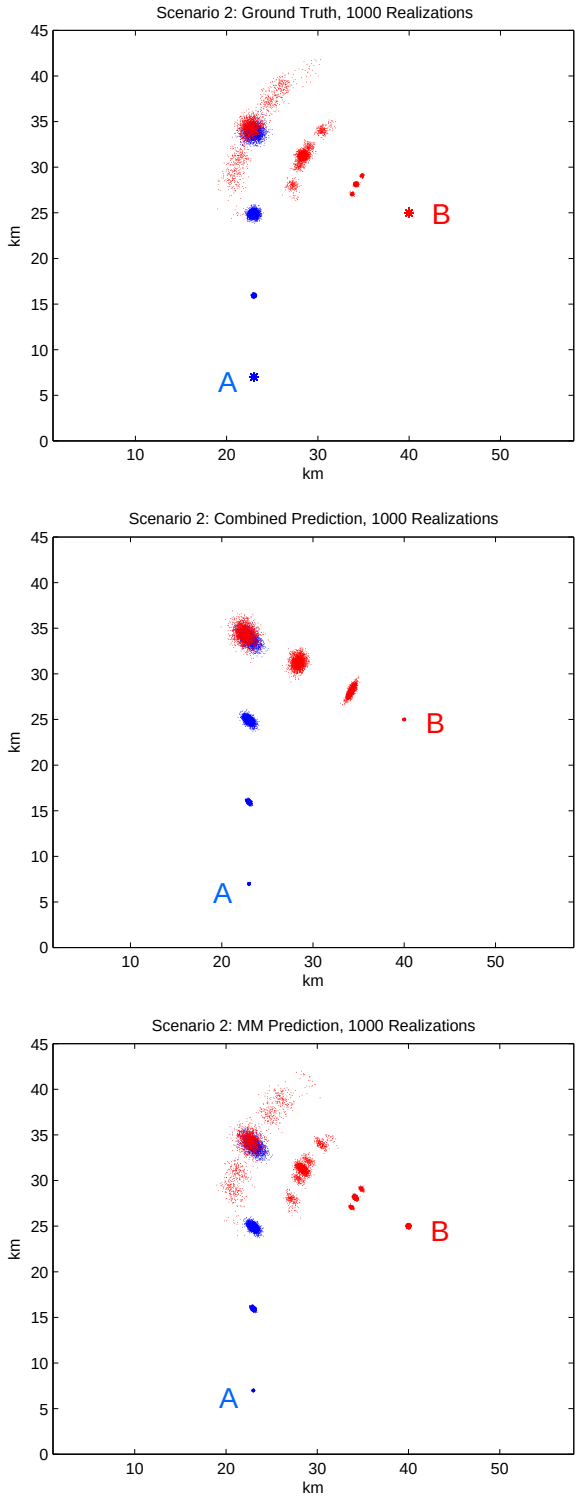

Figure 3.2: Scenario 2: Truth (top), Combined Prediction (middle) and MM Prediction (bottom)

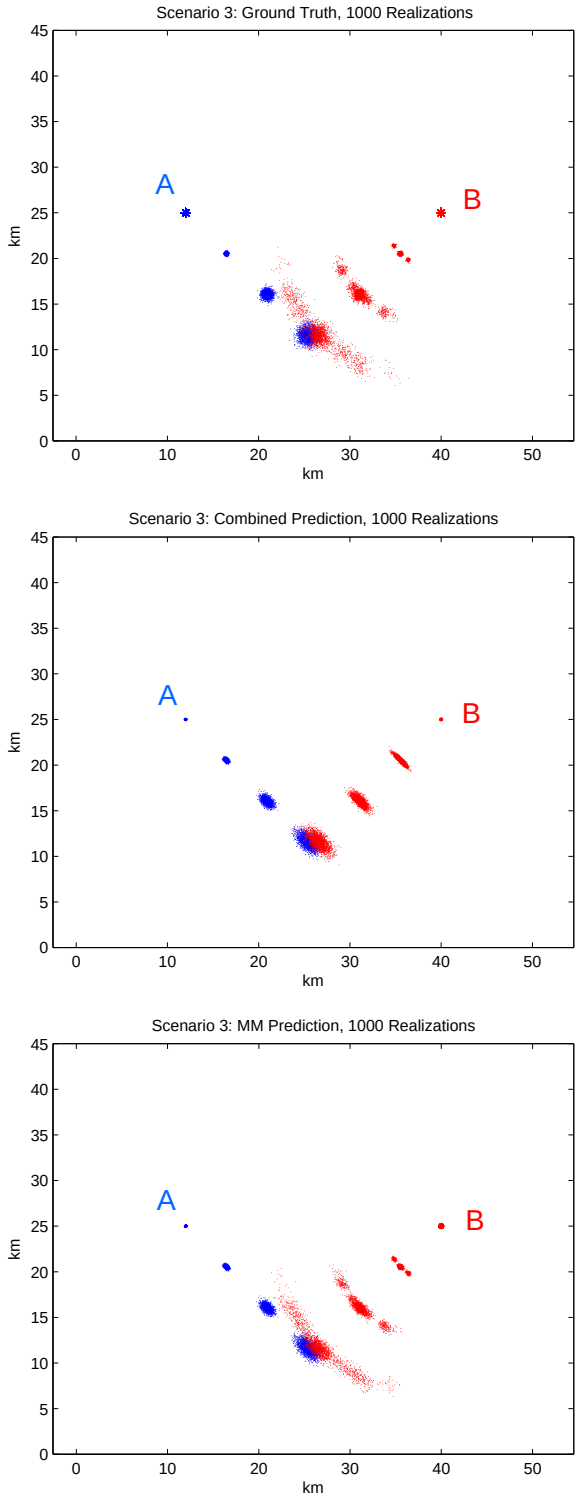

Figure 3.3: Scenario 3: Truth (top), Combined Prediction (middle) and MM Prediction (bottom)

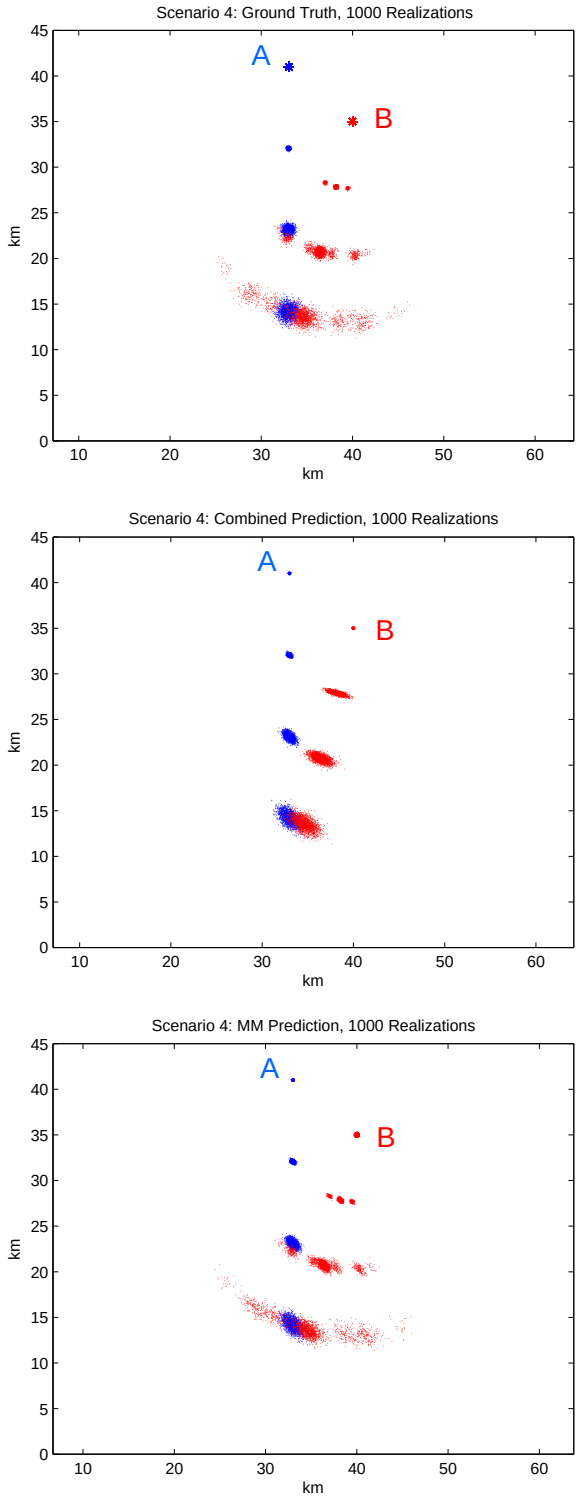

Figure 3.4: Scenario 4: Truth (top), Combined Prediction (middle) and MM Prediction (bottom)

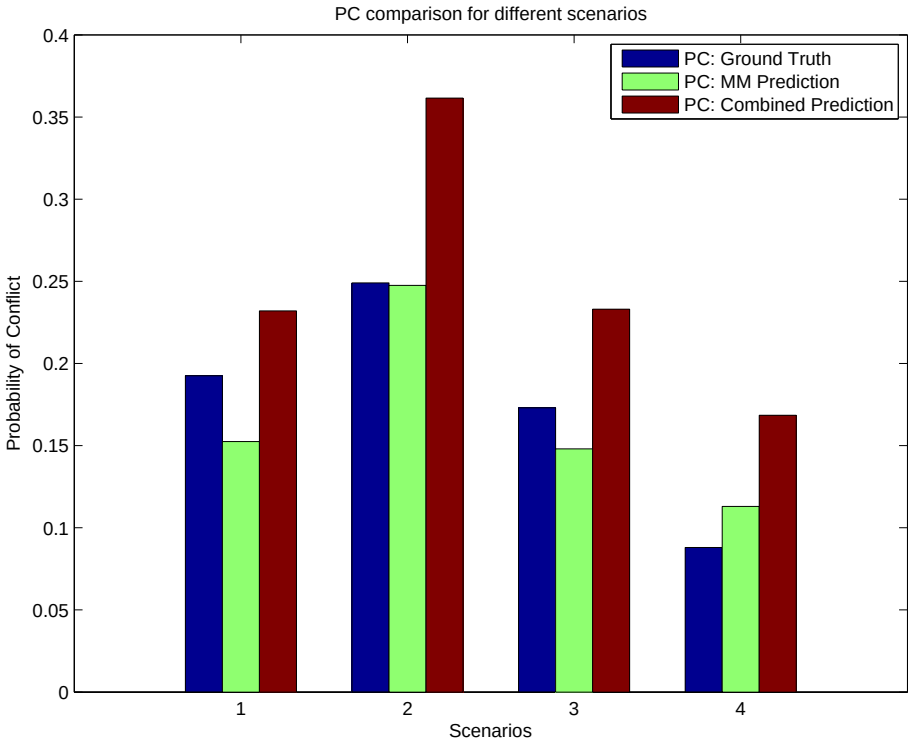

Figure 3.5: PC Comparison over four scenarios. MM prediction denotes the proposed (GM-based) PC prediction method, Combined Prediction denotes an existing (Gaussian-based) PC prediction method.

Chapter 4

Conflict Resolution (CR)

The MM formulation for CR and the optimal constrained sequential list Viterbi algorithm presented in this chapter were proposed in our paper [17].

4.1 When is CR Needed?

It is assumed that, at each time k (current time), all filtered estimates $\hat{x}^A_{k|k}$ and $\hat{x}^{B_b}_{k|k}$, $b = 1, 2, \ldots$, are available. The CD strategy, detailed in Section 3.1, is used to predict the GM PDFs of all intruders and, given the own-ship mode sequence $\mathfrak{M}_k^{k+N}$, compute the predicted $PC^{(b)}_{k+n}$, $n = 1, \ldots, N$; $b = 1, 2, \ldots$, where N is the look-ahead time horizon and b denotes intruder B^b.

A conflict alert is triggered "ON" iff

$$\max_{(b, 1 \le n \le N)} PC^{(b)}_{k+n} \ge \delta \tag{4.1}$$

where δ is a threshold.

Upon a conflict alert, the current mode sequence $\mathfrak{M}_k^{k+N}$ of the own-ship needs to be updated to provide safety (i.e., to guarantee $\max_{(b,1\le n\le N)} PC^{(b)}_{k+n} < \delta$ along the updated flight path).

4.2 Constrained Optimization: Maneuvering Cost Function

Let $c_{k+n}(m^{(i)}, m^{(j)})$, $i, j \in \{1, 2, \ldots, M\}$ be the cost of switching from mode $m^{(i)}$ at time $k + n - 1$ to mode $m^{(j)}$ at time $k + n$, $n = 1, 2, \ldots, N$. Then, the collision avoidance (CA) problem is formulated as the following "chance-constrained" stochastic MPC problem:

Minimize:

$$J(\mathfrak{M}_k^{k+N}) = \sum_{n=1}^{N} c_{k+n}(m^{(i_{k+n-1})}, m^{(i_{k+n})}) \tag{4.2}$$

subject to:

$$PC_{k+n}^{(b)} < \delta,\ n = 1, \ldots, N;\ b = 1, 2, \ldots \tag{4.3}$$

It is clear that the formulation of the total cost, given by (4.2), has some limitations. It only takes into account the costs for switching between flight modes and does not include any information about the own-ship's destination. The lack of destination information in the objective function leaves the CDR problem "half-solved" in a way because some extra guidance algorithm is required to drive the own-ship to its destination in the absence of conflicts. Likewise, based on (4.2) alone, the own-ship is not capable of returning to the originally intended flight plan or flight altitude. Although, careful design of the cost matrix may allow the own-ship to partially control its trajectory. The one property that is guaranteed as a result of (4.2) and (4.3) is that the overall trajectory of the own-ship will be conflict-free.

Despite these limitations, a strong argument still supports this formulation. First, the primary goal in CDR is to guarantee the safety of the aircraft involved in a close encounter. Trajectory optimization comes second meaning that only once a conflict is resolved will the own-ship be allowed to return to the original flight path or be re-routed to the next waypoint.

More terms could be added to the objective function to remedy such limitations. The only problem is that having more terms that need to be optimized can lead to drastic increases in the complexity of the resulting stochastic optimal control problem which can require extensively more computational resources. For many practical applications, e.g., UAV SA, the computational resources are quite limited. From this viewpoint, the formulation and approach, proposed in [17], that is presented in this chapter is quite attractive.

4.3 Viterbi Algorithm (VA) & Sequential List VA (SLVA)

The unconstrained optimization problem (4.2) is one of finding a best (least costly) path through a trellis digraph. Fig. 4.1 illustrates a typical model switching trellis (at current time k) with $M = 5$ models (nodes) and look-ahead time horizon $N = 6$ (i.e., ending time is $k+N$). Without loss of generality, it is assumed that at the start and end times (k and $k + N$, respectively) the own aircraft is in flight mode $m^{(1)}$, CV motion. As discussed later in Section 4.5), a complete trellis with $m^{(3)},\ m^{(5)}$ being right CT models and $m^{(2)},\ m^{(4)}$ being their symmetrical left CT models is used. The depicted trellis is not complete because switching from a left CT to a right CT (and vise versa) is only possible through the CV model.

Finding an unconstrained best path through a trellis can be easily and efficiently done via Dynamic Programming, e.g., via the popular Viterbi algorithm (VA) [11]. The constrained problem (4.2)–(4.3) is more complicated. The obvious idea of searching for a best path, after determining the feasible (conflict-free) trellis (a subset of the complete trellis) first is a dead end because it needs evaluation of PC over all

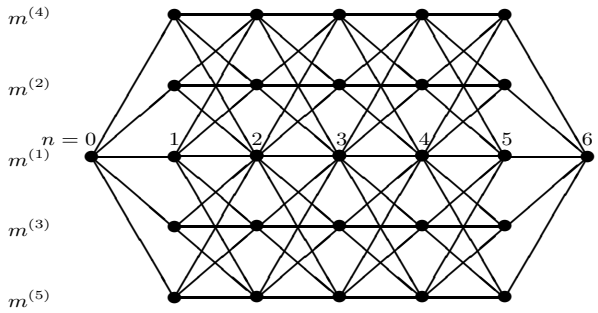

Figure 4.1: Model Switching Trellis with $M = 5$ and $N = 6$.

paths. Since the PC evaluation is the computational bottleneck in this problem, a better idea is to organize the search of the best paths in a sequential manner (in an increasing order of costs) and then check feasibility so that PC is only evaluated over lower-cost candidate paths. To put it simply, the strategy is to find the best path and check its feasibility. If it is not feasible, then the second best path is found, and its feasibility is checked. If the second best path is infeasible, the search continues for the third best path and so on and so forth until a path is found which is feasible.

The problem of finding the L best paths through directed graphs has been well studied in the area of computational geometry and many general algorithms exist. More efficient algorithms, particularly tailored to the special case of trellis graphs are available in the communications literature, [26,33,36], where algorithms for finding the L best paths through a trellis are generally referred to as the List Viterbi Algorithms (LVAs). For the purpose of finding an optimal solution to the constrained problem (4.2)–(4.3), in principle, any LVA can be used. However, the sequential LVA (SLVA) [36] best suits the problem at hand for two reasons: 1) it finds the next best path recursively based on the previous best paths, and 2) the search is organized via forward VA-like passes through the trellis, which allows already computed PCs to be reused along the conflict-free parts of future VA passes.

For simplicity, let the trellis states be denoted by $i \in \{1, 2, \ldots, M\}$ (instead of the previous MM notation $m^{(i)} \in \mathbb{M}$). The initial state of the trellis at time $n = 0$ is assumed to be 1. Let $\lambda_n^{(i)}$ be the minimum cost to reach state i at time n (from the known state 1 at time $n = 0$), and $j_n(i) \in \{1, 2, \ldots, M\}$ be the state occupied at time $n - 1$ by the best path into state i at time n.

In SLVA, the lth best path $p^{(l)}$ is found in a sequential manner based on the previously found $(l-1)$ best paths $p^{(1)},\ p^{(2)}, \ldots,\ p^{(l-1)}$ which are sorted in a non-decreasing order of costs.

$$p^{(l)} = \text{SLVA}(p^{(1)}, p^{(2)}, \ldots p^{(l-1)}),\ l = 2, 3, \ldots \tag{4.4}$$

where SLVA symbolizes one recursive step of the algorithm. The first step of SLVA is to obtain the best path, $p^{(1)}$, via VA. VA is explained in detail in Algorithm 2. Then, using $p^{(1)}$, $p^{(2)}$ can be found by mutating $p^{(1)}$ with only one forward, cheapest cost split point. If any other split points occur when mutating $p^{(1)}$ into $p^{(2)}$, then $p^{(2)}$ cannot be the next best path because it will have a higher cost than a path that has only one, cheapest cost split point from $p^{(1)}$. Furthermore, an issue may arise where multiple paths mutate from a previous best path with the same cost increase making all of them valid candidates to be the next best path. This issue is solely governed by the costs assigned between the nodes in the trellis.

So, if i_n^* is the state occupied by $p^{(1)}$ at time n, the forward cost-to-go $\lambda_n^{(i_n^*)}(2)$ of $p^{(2)}$ can be written as

$$\lambda_n^{(i_n^*)}(2) = \min\{(\lambda_{n-1}^{(i_{n-1}^*)} + c_n(i_{n-1}^*, i_n^*)), \min_{0\le j\le M,\ j\neq i_n^*}(\lambda_{n-1}^{(j)} + c_n(j, i_n^*))\} \tag{4.5}$$

The first term in (4.5) represents the 2nd best path to i_n^* which has merged to $p^{(1)}$ no later than time $n-1$. The second term represents the 2nd best path to i_n^* which has merged to $p^{(1)}$ no earlier than time n. In the forward pass of the algorithm, the one with the minimum cost remains in contention to become $p^{(2)}$, and the time of the last best merge, n_m, is recorded. Then, $p^{(2)}$ is the 2nd best path to time $n_m - 1$ as determined by the above recursion and is equal to $p^{(1)}$ from time n_m until the end. Finding $p^{(3)}$ from $p^{(1)}$ and $p^{(2)}$ can be organized in a similar manner. A formal, more detailed description of the algorithm is given in [36].

Algorithm 2 Viterbi Algorithm

Initialization:

for $i = 1$ to M **do**
  $\lambda_1^{(i)} = c_1(1, i)$
  $j_1(i) = 1$

Recursion:

for $n = 2$ to $N-1$ **do**
  for $i = 1$ to M **do**
    $\lambda_n^{(i)} = \min_{1\le j\le M}(\lambda_{n-1}^{(j)} + c_n(j, i))$
    $j_n(i) = \arg\min_{1\le j\le M}(\lambda_{n-1}^{(j)} + c_n(j, i))$

Termination:

$\lambda_N^{(1)} = \min_{1\le j\le M}(\lambda_{N-1}^{(j)} + c_N(j, 1))$
$j_N(1) = \arg\min_{1\le j\le M}(\lambda_{N-1}^{(j)} + c_N(j, 1))$

Backtracking:

$i_N^* = 1$
for $n = N-1$ to 1 **do**
  $i_n^* = j_{n+1}(i_{n+1}^*)$

Best Sequence:

$(1, i_1^*, \ldots, i_{N-1}^*, 1)$

4.4 Constrained Sequential List Viterbi Algorithm (CSLVA)

For this CDR method, CD is done by the PC prediction method discussed in Chapter 3 and simply implements (4.1). CR is done by the SLVA search strategy that was discussed in the previous section. A straightforward implementation of the standard SLVA search strategy which is where PC is computed for the entire time horizon of each next best path $p^{(l)}$ would lead to many duplicate computations of the same PC because $p^{(l)}$ has portions of itself that are also present in its children $(p^{(l+1)}, p^{(l+2)}, \ldots)$. To remedy this careless misuse of computational resources, two efficiency improvements are added to SLVA. By storing the split times and conflict times of subsequent paths as the search runs, the same PC will never be computed twice, and some next best paths can even be completely skipped over without computing any PCs at all because it will be already known that they are infeasible. These two efficiency improvements are illustrated in Figure 4.2, and Algorithm 3 shows the step-by-step algorithm for the CDR method.

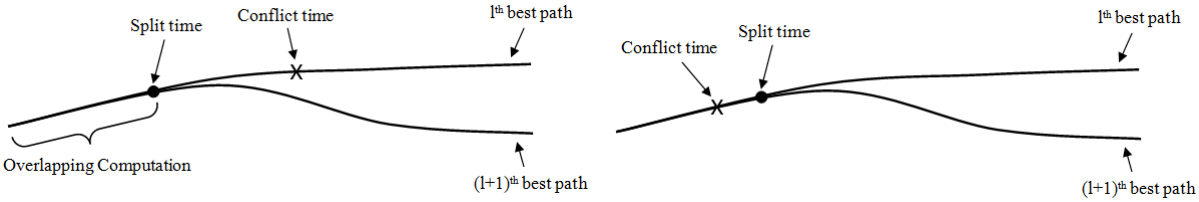

Figure 4.2: CSLVA Efficiencies. PC computations for the overlapping part of the child and parent path can be waived (top). The next best path $p^{(l+1)}$ can be completely skipped over without any PC computations (bottom).

Because the CDR method in this chapter operates off of (4.2) which lacks information about the own-ship's destination, an extra algorithm is executed in the absence of conflicts to reroute the own-ships's trajectory towards its destination. This rerouting algorithm is simple and intuitive but not efficient. It is basically an unconstrained SLVA search where each next best path that is found is linearly projected outwards for a fixed distance in order to "search" for the destination. The first path in the SLVA search that is found to have a point that lies within a specified distance threshold of the destination's position is chosen as the rerouted trajectory, and the rerouted trajectory is terminated at the point that "hit" the destination. The computational cost of this rerouting method depends on many factors such as the number of discrete modal states, length of the look-ahead time horizon, turning limitations of the own-ship, and geometry between the own-ship and destination at the time of rerouting. Fig. 4.3 illustrates this rerouting method and shows what paths within the trellis would be considered acceptable for rerouting the own-ship to its depicted waypoint.

Algorithm 3 Proposed CDR Algorithm

At time k:

Initialization:

$l = 1$

$p^{(1)} = (1, 1, \ldots, 1)$

Conflict Detection:

for $n = 1$ to N **do**

  for b = 1,2,... **do**

    Compute $PC_{k+n}^{(b)}(p^{(l)})$

    if $PC_{k+n}^{(b)}(p^{(l)}) \geq \delta$ **then**

      Go to 10:

$k = k + 1$

Go to 1:

Conflict Resolution: (Search trellis for best CR maneuver sequence)

a) Find conflict times:

for b = 1,2,... **do**

  $n_c^{(b)}(p^{(l)}) = \min_{1 \leq n \leq N} \{n : PC_{k+n}^{(b)}(p^{(l)}) \geq \delta\}$

b) Find next best path & split times:

$l = l + 1$

$p^{(l)} = \text{SLVA}(p^{(1)}, p^{(2)}, \ldots, p^{(l-1)})$

for $j = 1$ to $l - 1$ **do**

  $n_s^{(l|j)} = \min_{0 \leq n \leq N} \{n : p^{(l)} \text{ diverges from } p^{(j)}\}$

c) Find largest split time from conflict-free parent subsequence:

$n_s^{(l|j_0)} = \min_{1 \leq j \leq l-1} \{n_s^{(l|j)} : n_s^{(l|j)} < n_c^{(b)}(p^{(j)})\}$

if $n_s^{(l|j_0)} = \emptyset$ **then**

  Go to 10:

for $n = n_s^{(l|j_0)}$+1 to N **do**

  for b = 1,2,... **do**

    Compute $PC_{k+n}^{(b)}(p^{(l)})$

    if $PC_{k+n}^{(b)}(p^{(l)}) \geq \delta$ **then**

      Go to 10:

CR sequence = $p^{(l)}$, recalculate guidance from state at time $k + N$ to next WP (or destination).

Go to 3:

4.5 Computational Efficiency Evaluation

The performance of the CR method, CSLVA, is compared, through simulation and data analysis, with a standard implementation of SLVA over SA UAV encounter scenarios that were previously described in Section 3.3. The results clearly show the feasibility of and computational efficiency improvements achieved by the CSLVA.

Two types of encounter scenarios are considered, one in which the own-ship makes strictly horizontal resolution maneuvers and one in which the own-ship makes strictly vertical resolution maneuvers. The horizontal scenarios are contained within the horizontal plane Oxy, and the dynamic model (2.1) of the own-ship A has $M = 5$ motion models which are $m^{(1)} = \text{CV}$ (straight) with $\omega^{(1)} = 0\,^{\circ}/s$, $m^{(2)} = \text{CT}$ (soft

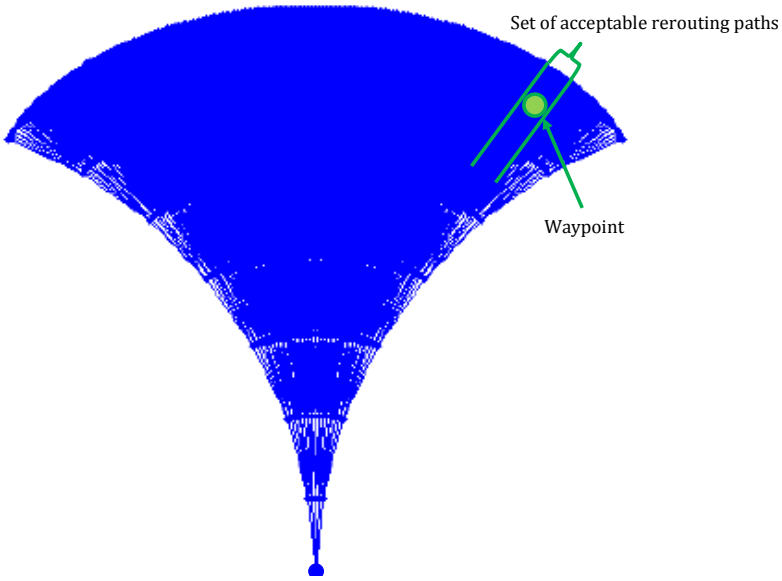

Figure 4.3: Unconstrained SLVA-based rerouting illustration.

left turn) with $\omega^{(2)} = 1\,^{o}/s$, $m^{(3)} = \text{CT}$ (soft right turn) with $\omega^{(3)} = -1\,^{o}/s$, $m^{(4)} = \text{CT}$ (hard left turn) with $\omega^{(4)} = 2\,^{o}/s$, and $m^{(5)} = \text{CT}$ (hard right turn) with $\omega^{(5)} = -2\,^{o}/s$, where ω is the turn rate.

The model transition costs used are

$$c(i,j) = |\omega^{(j)} - \omega^{(i)}|,\ i,j = 1,\ldots,5 \tag{4.6}$$

where $m^{(1)}$ is the initial and final trellis state as seen in Fig. 4.1. The look-ahead time horizon is $N = 10$ for all scenario types in this section.

Each intruder B is modeled through the same hybrid system with the same initial and transition probabilities that is described in Section 3.3. The time step duration is $T = 5\,\text{sec}$, and the process noise covariance and number of MC runs are the same as in Section 3.3. The minimum horizontal separation distance that defines a conflict is $\lambda_{\text{xy}} = 5km$, and the threshold on PC for triggering a conflict alert is $\delta = 10^{-3}$.

Figs. 4.4 and 4.5 show two scenarios with one intruder (top) and two intruders (bottom), respectively. Integer labels indicate time steps. The uncertainty in intruders' trajectories is illustrated by scatter plots sampled from the predicted mixtures. At the time step in the future that a conflict is deemed to occur, the predicted density of the intruder that is involved in the conflict is highlighted red. The blue dashed line shows the own-ship's desired path to its next waypoint, and the blue solid line shows the minimum cost, conflict-free path computed by the proposed CR algorithm. The figures illustrate the capability of the algorithm to

successfully resolve a conflict by finding a minimum cost horizontal maneuver. Each scenario was repeated 5000 times with random horizontal maneuver sequences of the intruder(s) and no conflict occurred with the own-ship minimum maneuver cost trajectory.

The vertical scenarios are contained within the vertical plane $O\mathsf{xz}$. The dynamic model (2.1) of the own-ship A has $M = 5$ motion models which are $m^{(1)} = \text{CH}$ (level cruise) with $v^{(1)} = 0m/s$, $m^{(2)} = \text{CD}$ (soft climb) with $v^{(2)} = 10m/s$, $m^{(3)} = \text{CD}$ (soft descent) with $v^{(3)} = -10m/s$, $m^{(4)} = \text{CD}$ (hard climb) with $v^{(4)} = 20m/s$, $m^{(5)} = \text{CD}$ (hard descent) with $v^{(5)} = -20m/s$, where v is the vertical velocity. The model transition costs for the vertical scenarios are the same as in (4.6) except with ω replaced by v.

Each intruder B has three motion models which are $m^{(1)} = \text{CH}$ (level cruise) with $v^{(1)} = 0m/s$, $m^{(2)} = \text{CD}$ (soft climb) with $v^{(2)} = 10m/s$, and $m^{(3)} = \text{CD}$ (soft descent) with $v^{(3)} = -10m/s$ with the same flight mode initial and transition probabilities as those used in the horizontal scenarios.

The process noise covariance is $Q = q^2 I$ with $q = 1m/s^2$, and the time step duration and number of MC runs are the same in the horizontal scenarios. The minimum vertical separation distance that defines a conflict is $\lambda_{\mathsf{z}} = 1000m$, and the threshold on PC for triggering a conflict alert is the same as in the horizontal scenarios.

Fig. 4.6 shows one scenario with one intruder (top) and two intruders (bottom), respectively. It illustrates the capability of the algorithm to successfully resolve a conflict by finding a minimum cost vertical maneuver. The scenario was repeated 5000 times with random vertical maneuver sequences of the intruder(s) and no conflict occurred with the own-ship minimum maneuver cost trajectory.

The computational efficiency of CSLVA as compared with a direct implementation of the standard SLVA is shown in Table 4.1. The column titled "Search Depth" shows the number of best paths that were sequentially found to be unsafe by both algorithms before finding the best conflict-free path. Because the only difference between the two algorithms is their computational efficiency and not their solution, the "Search Depth" number is the same for both algorithms. Columns "# PC Checks" show the total number of instantaneous PCs that were computed by each algorithm, respectively. Columns "Comp. Time" show the amount of time that each algorithm took to find the best conflict-free path. Column "Speedup" shows the ratio of computation times of SLVA and CSLVA, given in the fifth and sixth columns, respectively.

In all scenarios, CSLVA was more efficient than the direct SLVA. This efficiency improvement is a result of dramatically reducing the number of PCs computed (illustrated by the comparison of columns three and four in Table 4.1). The amount of improvement is scenario dependent. For the more difficult horizontal scenarios, the speedup can be quite significant (as high as 4.83 times), but, for the much easier vertical scenarios, the speedup is not that significant (can be as low as 1.12 times). In conclusion, the computational savings are directly proportional to the difficulty of the scenario.

Table 4.1: Computational Performance: CSLVA vs. SLVA

Scenario ID	Search Depth	# PC Checks		Comp. Time (sec)		**Speedup** (times)
		SLVA	CSLVA	SLVA	CSLVA	
H1:1	7433	59265	1424	12.09	2.50	**4.83**
H1:2	28102	227152	3796	38.66	8.38	**4.61**
H2:1	3274	20864	368	5.69	1.29	**4.41**
H2:2	3275	21003	379	5.61	1.80	**3.11**
V:1	361	5073	1473	1.79	1.59	**1.12**
V:2	1832	26057	7795	5.54	3.51	**1.57**

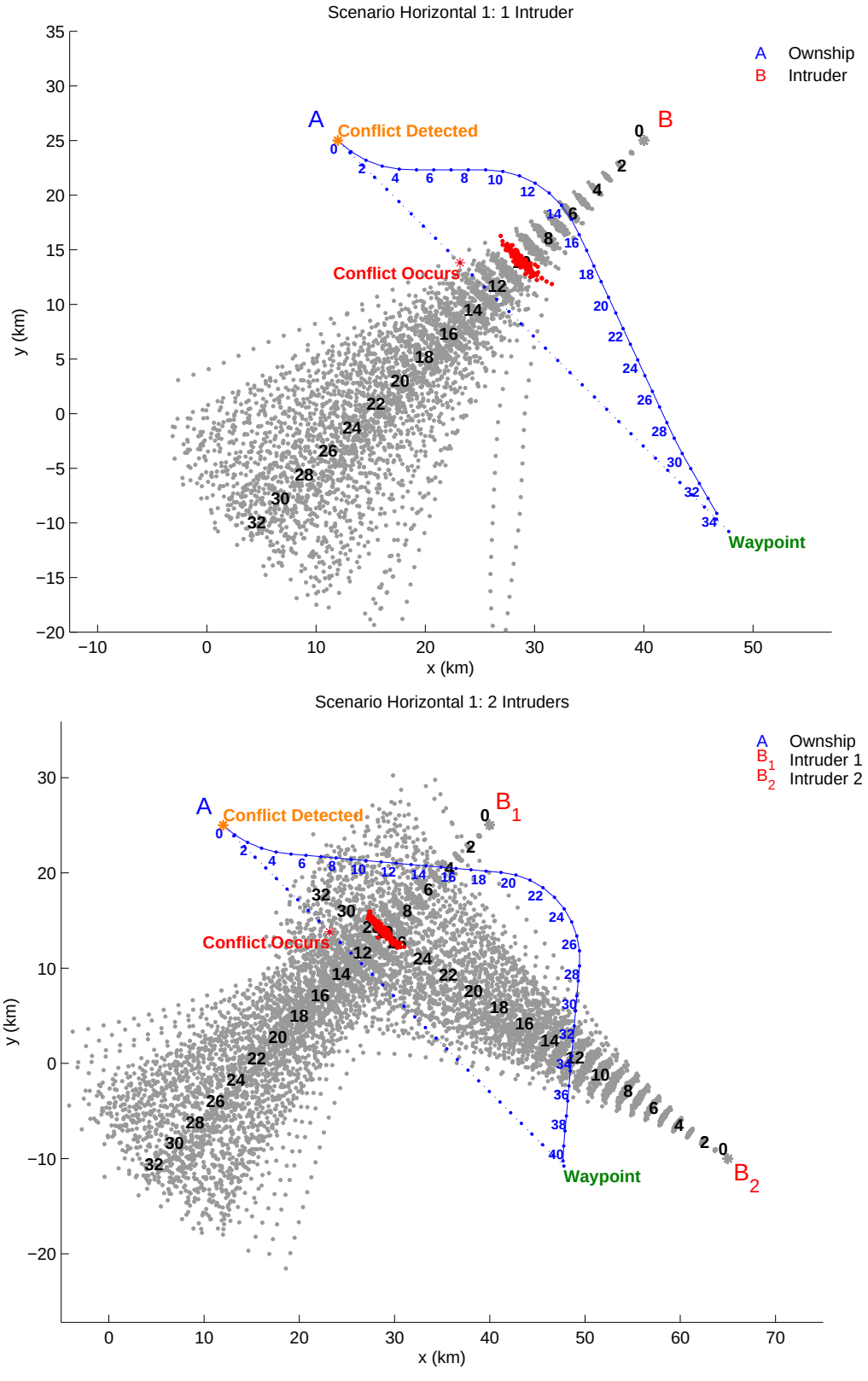

Figure 4.4: Horizontal Scenario (H1): One Intruder (top), Two Intruders (bottom).

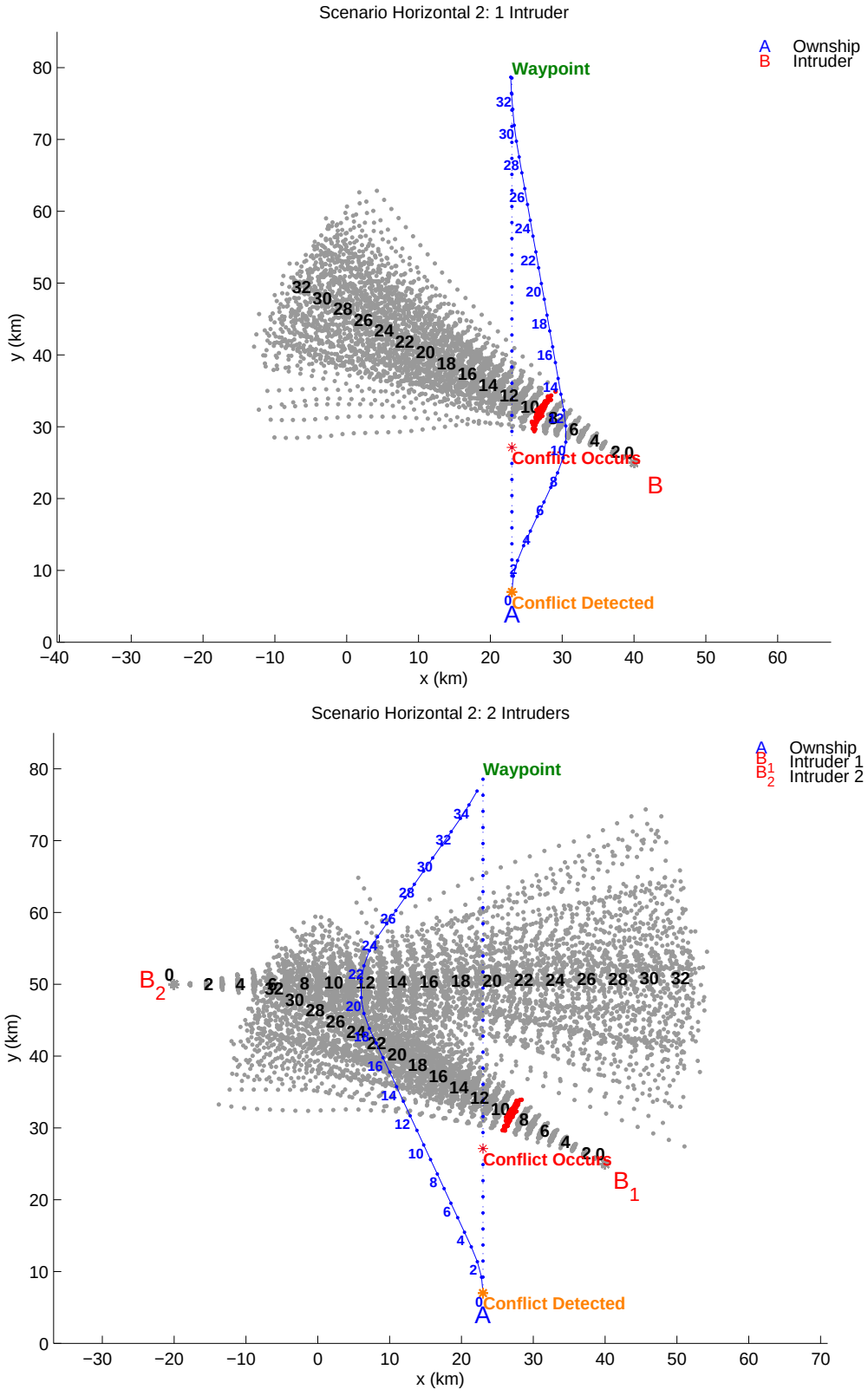

Figure 4.5: Horizontal Scenario (H2): One Intruder (top), Two Intruders (bottom)

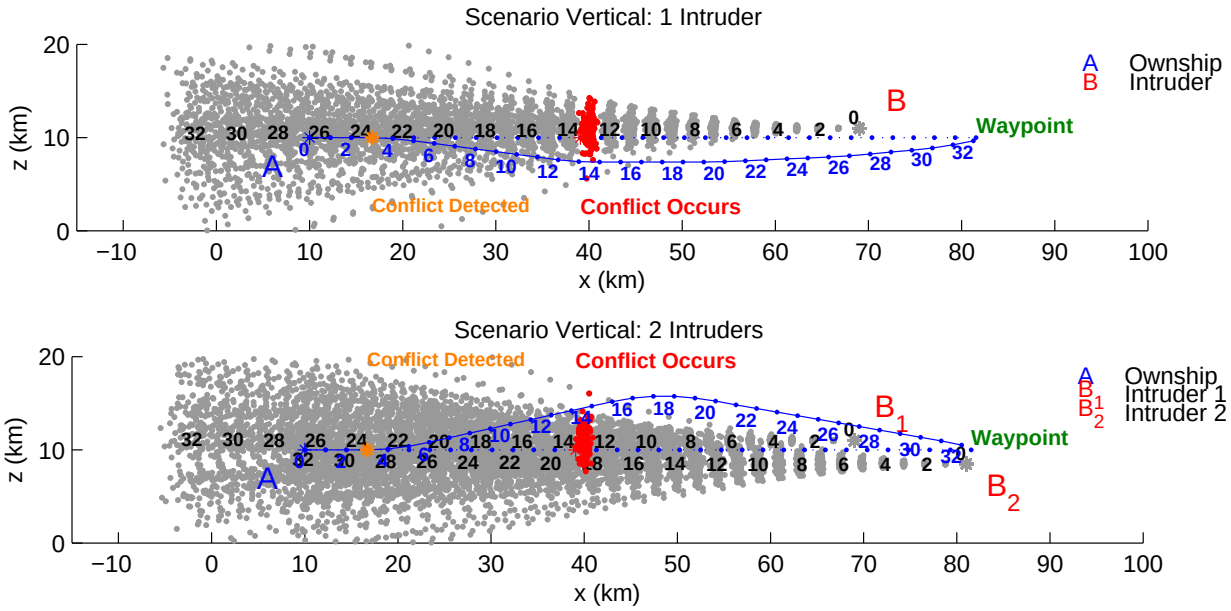

Figure 4.6: Vertical Scenario (V): One Intruder (top), Two Intruders (bottom).

Chapter 5

Enhanced CR

The enhanced CR method described in this chapter was proposed in our paper [15].

5.1 Constrained Optimization: Enhanced Cost Function

Extending the formulation of the previously discussed maneuvering cost function, let ζ be the travel cost for A per unit distance, p_{k+N} be the position vector of A at time $k+N$, $\bar{p}^{(0)}_{k+N|k}$ be the predicted position in the current flight plan, and d be the position vector of the next destination. Now, the CA problem is formulated as the following, more robust "chance-constrained" stochastic MPC problem:

Minimize:

$$J(\mathfrak{M}_k^{k+N}) = \sum_{n=1}^{N} c_{k+n}(m^{(i_{k+n-1})}, m^{(i_{k+n})}) + \zeta(E(\|p_{k+N} - d\|) - |\bar{p}^{(0)}_{k+N|k} - d|) \tag{5.1}$$

subject to:

$$PC^{(b)}_{k+n} < \delta,\ n = 1, \ldots, N;\ b = 1, 2, \ldots \tag{5.2}$$

The new objective function (5.1) is more general and more powerful than (4.2) in that the destination is now explicitly incorporated. The objective function in (4.2) guarantees a conflict-free path but not a path that ends up at or near the destination. Now, with the added extra term $\zeta(E(\|p_{k+N} - d\|) - |\bar{p}^{(0)}_{k+N|k} - d|)$, the optimal trajectory will naturally arrive at or very near to its intended destination. For scenarios involving a vertical maneuver, the destination is an altitude level rather than a coordinate in the horizontal plane.

(4.3) and (5.2) are exactly the same, and here is why. A very large cost for violating minimum safety separations could be added to the objective function instead of having this violation as a hard constraint. However, because safety is the absolute number one priority in this problem, fulfilling the safety requirements must be strictly enforced and not left to be relaxed. If violating minimum safety separations was considered as some penalty cost in the objective function instead of a hard set constraint, some scenario may exist where

two aircraft become fatally close to one another in the act of trying to minimize the total cost of the objective function. Thus, as proposed in [15], the overall problem is to find the least expensive destination-directed trajectory that guarantees the specified safety level.

5.2 CSLVA-directed Search

Without the extra term $\zeta(E(\|p_{k+N} - d\|) - |\bar{p}^{(0)}_{k+N|k} - d|)$ in (5.1), an optimal solution can be efficiently found using CSLVA as discussed in Chapter 4. Including this extra term in the cost function, however, makes the optimization problem a lot harder. The total cost is no longer a sum of the transitional costs over the trellis, and CSLVA is no longer optimal. Nevertheless, it is still possible to take advantage of CSLVA and organize an efficient search over the trellis to minimize (5.1). Here, the suboptimal algorithm, referred to as CSLVA-directed search, is discussed in detail.

CSLVA-directed search minimizes the two term objective function by initially finding the first best path through the trellis minimizing the first term of (5.1), the mode switching costs. The divergence from destination cost, second term of (5.1), of the first best path is added to the mode switching costs to get a total cost. The total cost of the first best path is then recorded. Next, the second best path through the trellis is found. The divergence from destination cost is computed, and the total cost of the second best path is recorded. The algorithm keeps sequentially searching for next best paths through the trellis and recording their computed total costs until the allotted time runs out. If an l^{th} best path is found to be infeasible during the search, then its total cost is neither computed nor recorded. Out of all of the feasible paths that were searched, the path with the lowest total cost is chosen as the minimizer of (5.1). The allotted search time cannot exceed the duration of a time step in order for the algorithm to be considered for real-time applications. Algorithm 4 shows the step-by-step process of CSLVA-directed search.

Algorithm 4 CSLVA-directed CR Search

At time k:

Initialize: $l = 1,\ p^{(1)} = (m_k^{(1)}, m_{k+1}^{(1)}, \ldots, m_{k+N}^{(1)})$
if $p^{(1)}$ is conflict-free **then**
  Record $J(p^{(1)})$
while current runtime $<$ time step duration **do**
  Record the earliest conflict time $n_c(p^{(l)})$ for $p^{(l)}$
  $l = l + 1$
  Find the next best path $p^{(l)}$
  Find the largest split time $n_s^{(l|j)}$ that $p^{(l)}$ has with any parent path $p^{(j)}$
  Recall the conflict time $n_c(p^{(j)})$
  if $n_c(p^{(j)}) > n_s^{(l|j)}$ or $n_c(p^{(j)}) = \emptyset$ **then**
    if $p^{(l)}$ is conflict-free from $n_s^{(l|j)}$ to $k + N$ **then**
      Record $J(p^{(l)})$
Execute action $m_k^{(i_k)}$ from $\arg\min_{p^{(l)}} J(p^{(l)})$

For numerical evaluation of the expectation $E(\|p_{k+N} - d\|)$ in (5.1), one can use Monte Carlo approximation via sampling random trajectories from (2.1) using the given path $\mathfrak{M}_k^{k+N}$. It is also efficient to replace $E(\|p_{k+N} - d\|)$ in (5.1) by $(\|E(p_{k+N}|z^k) - d\|)$ and take $E(p_{k+N}|z^k)$ from $\hat{x}_{k+N|k}^{(i_{k:k+N})}$ which is already computed by the trajectory prediction part of the CDR algorithm. For this implementation and the other two described next, the latter is used in the simulations.

5.3 SMC-based Search

Typically, sequential Monte Carlo has been used for the problem of state estimation and is generally labeled as particle filtering. Only recently has SMC been used for solving MPC optimization problems [9, 18, 24]. A non-convex, non-linear MPC problem can have several local minima or maxima. Therefore, using an optimization method that produces globally optimum solutions such as SMC, a form of simulated annealing, is critical. SMC optimization can also be thought of as a genetic ("survival of the fittest") type of algorithm where a particle population evolves based on some selection and mutation criteria. An SMC-based search works by sampling particles from an importance distribution and iteratively resampling these particles until the particle population has matured enough to produce a satisfactory estimate of the global optimizer or until some stopping criterion has been met.

Algorithm 5 SMC-based CR Search

At time k:

Initialize: $l = 1, n = 0$
while $n \leq N_p$ **do**
  Sample a path from the random walk directory
  if the sampled path is conflict-free **then**
    $p_n^{(l)}$ = sampled path
    Record the total cost $J(p_n^{(l)})$ and weight $w(p_n^{(l)})$
    $n = n + 1$
  Remove the sampled path from random walk directory
while the allotted runtime has not run out **do**
  $l = l + 1$
  $n = 0$
  while $n \leq N_p$ **do**
    Resample a path $p_{n_0}^{(l)}$ from the previous generation $p^{(l-1)} = (p_1^{(l-1)}, p_2^{(l-1)}, \ldots, p_N^{(l-1)})$ where the probability to resample a path is $\frac{w(p_{n_0}^{(l-1)})}{\sum_{n=1}^{N} w(p_n^{(l-1)})}$
    if $p_{n_0}^{(l)}$ is conflict-free **then**
      $p_n^{(l)} = \text{mutate}(p_{n_0}^{l})$
      Record the total cost $J(p_n^{(l)})$ and weight $w(p_n^{(l)})$
      $n = n + 1$
Execute action $m_k^{(i_k)}$ from $\arg\min_{p_n^{(l)}} J(p_n^{(l)})$

With respect to the considered CDR problem, a particle is a maneuver sequence. In the SMC-based

implementation shown in Algorithm 5, the initial particle population of size N_p is sampled from a random walk directory. The total cost of each initial particle, mode switching costs plus divergence from destination cost, is computed and recorded. Each particle's weight is computed based on its total cost and then recorded. The initial particle population is not allowed to contain duplicates of the same particle. Therefore, after a particle has been sampled, it is removed from the random walk directory so that it cannot be sampled again in a future iteration of the initialization process. If a sampled path violates the PC constraint, then no data is recorded for that path. The initialization process continues until the initial particle population has N_p particles. Then, resampling begins wherein particles are resampled based on their normalized weights. After being resampled, a mutation is introduced to the particle (i.e., the maneuver sequence is slightly altered). The total costs and weights of the second generation particles are computed and recorded, and the third generation particle population is formed by resampling and mutating the second generation particles. Recurring epochs of resampling continue until the allotted time runs out at which point the estimate of the global minimizer is chosen from the final particle population.

5.4 Exhaustive Search

To be able to accurately compare the optimization capabilities of the CSLVA-directed and SMC-based search methods, for each simulated scenario, the optimal solution is found by implementing a brute force (BF) exhaustive search which enumerates over all possible maneuver sequences. Throughout the search, the cheapest path found so far is retained. Each path is first checked for feasibility. If a path is found to be infeasible at some time n_0, then it is dropped immediately from further consideration along with all paths that stem from it starting at n_0. If a path is feasible, its total cost is computed and compared with the total cost of the cheapest path found so far, and the best of them is retained.

The computational burden of a brute force search depends on the length of the time horizon and the number of discrete modal states. For example, if a time horizon of 10 time steps with 5 modal states is used, approximately 9.7 million paths need to be assessed which could take upwards of several hours. However, if a time horizon of just 6 time steps with 5 modal states is used, only approximately 15.6 thousand paths need to be assessed which could take a reasonable one to two minutes. Since CDR must be executed in a very short time frame, the stopping criterion for the CSLVA-directed and SMC-based searches in the simulation is set to 10 seconds which is the duration of a single time step.

5.5 Trajectory Optimality Evaluation

In Section 4.5, the computational efficiency of CSLVA (with the maneuvering cost function) was demonstrated. In this section, the optimality of the solution produced by CSLVA (with the enhanced cost function)

is compared against the solution produced by the popular SMC CR method that is found in the literature. Comparing the optimality of these two methods requires the optimal solution which is found by a brute force (BF) exhaustive search. SA UAV encounter scenarios are considered here just as in all other simulation sections.

This section considers the same two types of aircraft-aircraft encounter scenarios, horizontal and vertical taken separately, as in Section 4.5 with the addition of an aircraft-weather horizontal encounter scenario. The aircraft-weather encounter scenario involves the own-ship avoiding bad weather regions and possibly intruders at the same time in the horizontal plane.

The horizontal scenarios are contained within the horizontal plane $O\mathsf{xy}$, and the dynamic models for the own-ship A and intruders B are exactly the same as in the horizontal scenarios of Section 4.5.

The "turning costs", or costs of being in a particular motion model at time k, are $c_k(m^{(1)}) = 1$ (straight), $c_k(m^{(2)}) = 5$ (soft left turn), $c_k(m^{(3)}) = 5$ (soft right turn), $c_k(m^{(4)}) = 9$ (hard left turn), $c_k(m^{(5)}) = 9$ (hard right turn). For all scenarios in this section, the look-ahead time horizon is $N = 6$, $\zeta = 2.5$, the time step duration is $T = 10\,\text{sec}$, the threshold on PC for triggering a conflict alert is $\delta = 10^{-2}$, and the process noise covariances and number of MC runs are the same as in Section 4.5. The minimum horizontal separation distance that defines a conflict is $\lambda_{\mathsf{xy}} = 9.26km$.

Figs. 5.1 and 5.2 show two scenarios with one intruder (top) and two intruders (bottom), respectively. The blue, red, and green trajectories are the minimum cost, conflict-free solutions produced by the brute force, SMC, and CSLVA methods, respectively. For all three trajectories, every fifth time step is enclosed by a shape such as a square, diamond, or circle. The uncertainty in intruders' trajectories is illustrated by scatter plots sampled from the predicted mixtures. The figures illustrate the capability of CSLVA-directed search to perform better than the SMC-based method in strictly horizontal aircraft-aircraft scenarios. All scenarios in this section were repeated 1000 times with random maneuver sequences for the intruder(s) and random spatial densities for the weather regions, and no conflict occurred with the own-ship minimum cost trajectory.

The vertical scenarios are contained within the vertical plane $O\mathsf{xz}$, and the dynamic models for the own-ship A and intruders B are exactly the same as in the vertical scenarios of Section 4.5.

The "altitude adjustment costs", or costs of being in a particular motion model at time k, are $c_k(v^{(1)}) = 1$ (level cruise), $c_k(v^{(2)}) = 3$ (soft climb), $c_k(v^{(3)}) = 2$ (soft descent), $c_k(v^{(4)}) = 5$ (hard climb), $c_k(v^{(5)}) = 4$ (hard descent). The minimum vertical separation distance that defines a conflict is $\lambda_{\mathsf{z}} = 4km$.

Fig. 5.3 shows one scenario with one intruder (top) and two intruders (bottom), respectively. The figure illustrates the capability of CSLVA-directed search to perform much better than the SMC-based method in V1_1 and slightly better in V1_2 which is the outlier in Fig. 5.5.

The weather scenarios are set up just like the horizontal scenarios except with bad weather regions added that the own-ship must avoid. The dynamic model for the own-ship and all simulation parameters are exactly the same as in the horizontal scenarios within this section. The Gaussian mixture used for the bad weather region is comprised of six Gaussian components. As can be seen in Fig. 5.4, some parts of the weather region are severe and, therefore, should be more strongly avoided by the own-ship. Whereas, some parts are mild enough that the own-ship can fly near or even through.

Fig. 5.4 shows one scenario with only weather (top) and one scenario with weather and one intruder (bottom), respectively. The figure illustrates the capability of CSLVA-directed search to perform better than the SMC-based search in strictly horizontal aircraft-weather scenarios.

The CSLVA-directed and SMC-based searches are compared in terms of the closeness of their solutions to the optimal (BF) solution over eight scenarios. For each scenario, the minimum cost achieved by each algorithm is shown in Fig. 5.5. All costs are normalized (divided by the corresponding optimal cost) to make the visual comparison clearer. In all scenarios, CSLVA-directed search achieves a minimum cost that is closer to the optimal than the minimum cost achieved by SMC-based search. Except for V1_2 (i.e., in all other seven scenarios), the cost achieved by CSLVA is pretty close (or identical, e.g., H2_2 and V1_1) to the optimal cost. Scenario V1_2 turns out to be difficult for both suboptimal searches, but CSLVA is still slightly better.

Because CSLVA searches the trajectory space in order from lowest turning cost trajectories to highest turning cost trajectories gradually, CSLVA is more capable of minimizing the first cost of (5.1) while also addressing the second term of (5.1). In contrast, the SMC algorithm does not have a direct way to minimize the turning cost because it searches the trajectory space in a random walk fashion. As a result, CSLVA-directed search produces resolution trajectories that are smoother and less jagged than resolution trajectories produced by the SMC algorithm.

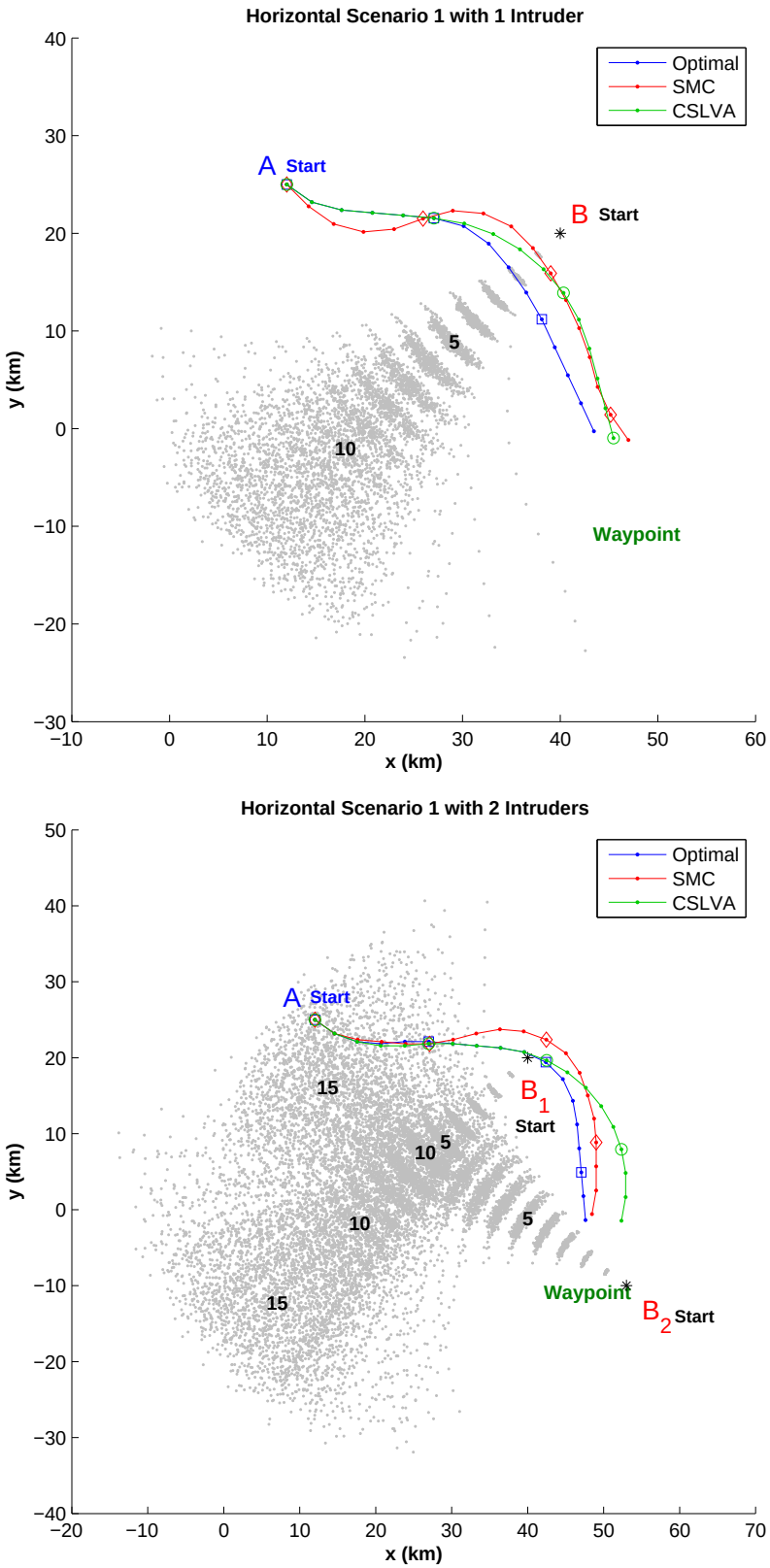

Figure 5.1: Horizontal Scenario 1: One Intruder (H1_1) (top), Two Intruders (H1_2) (bottom).

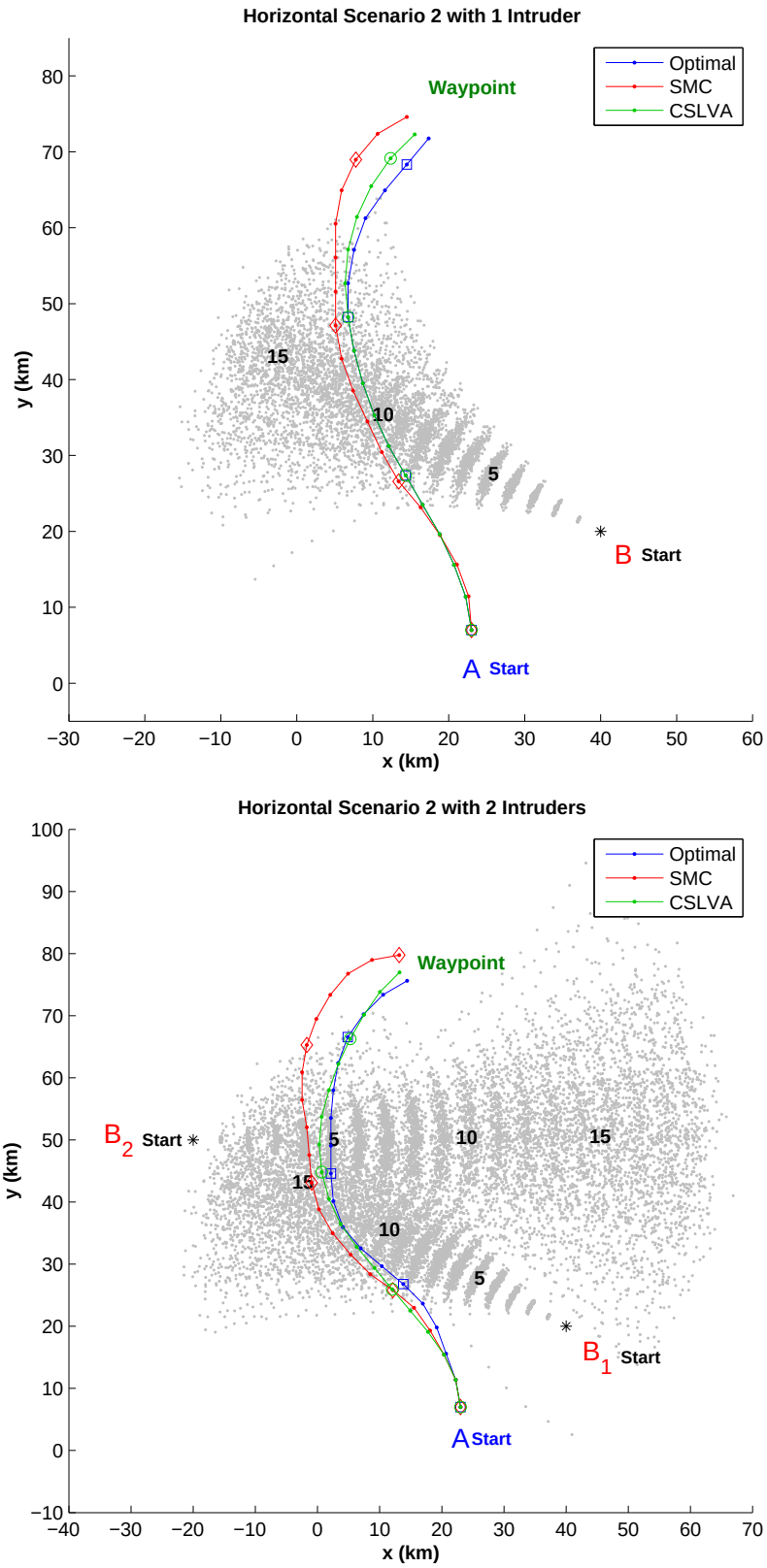

Figure 5.2: Horizontal Scenario 2: One Intruder (H2_1) (top), Two Intruders (H2_2) (bottom).

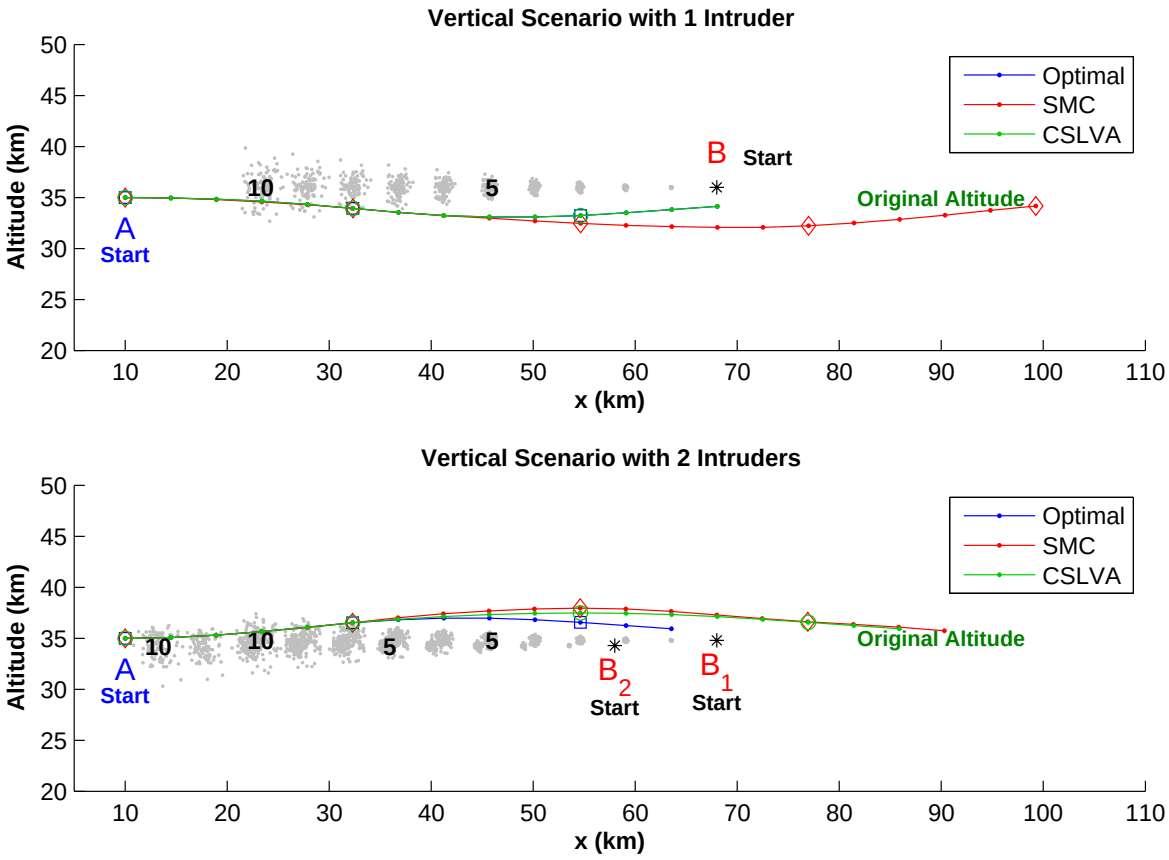

Figure 5.3: Vertical Scenario: One Intruder (V1_1) (top), Two Intruders (V1_2) (bottom).

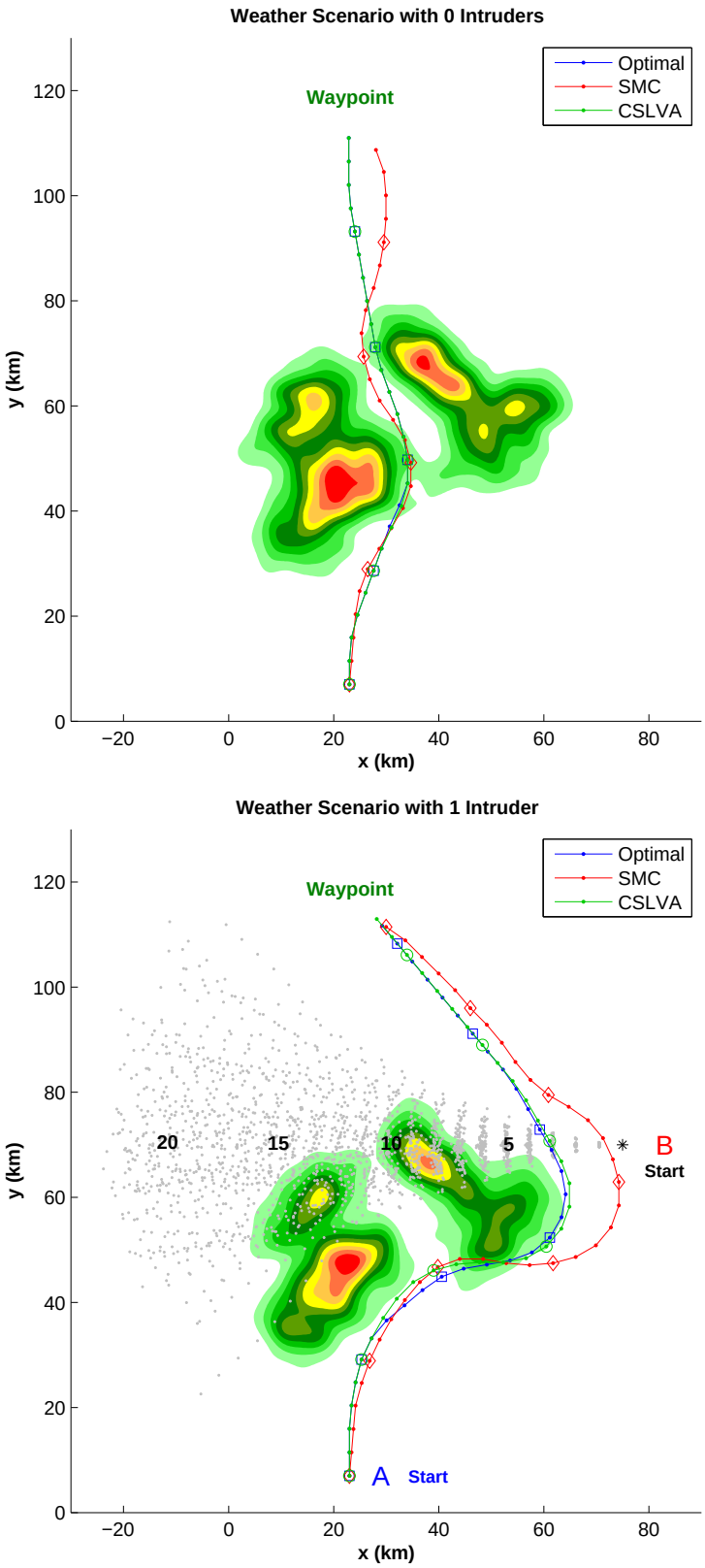

Figure 5.4: Weather Scenario: No Intruders (W1_0) (top), One Intruder (W1_1) (bottom).

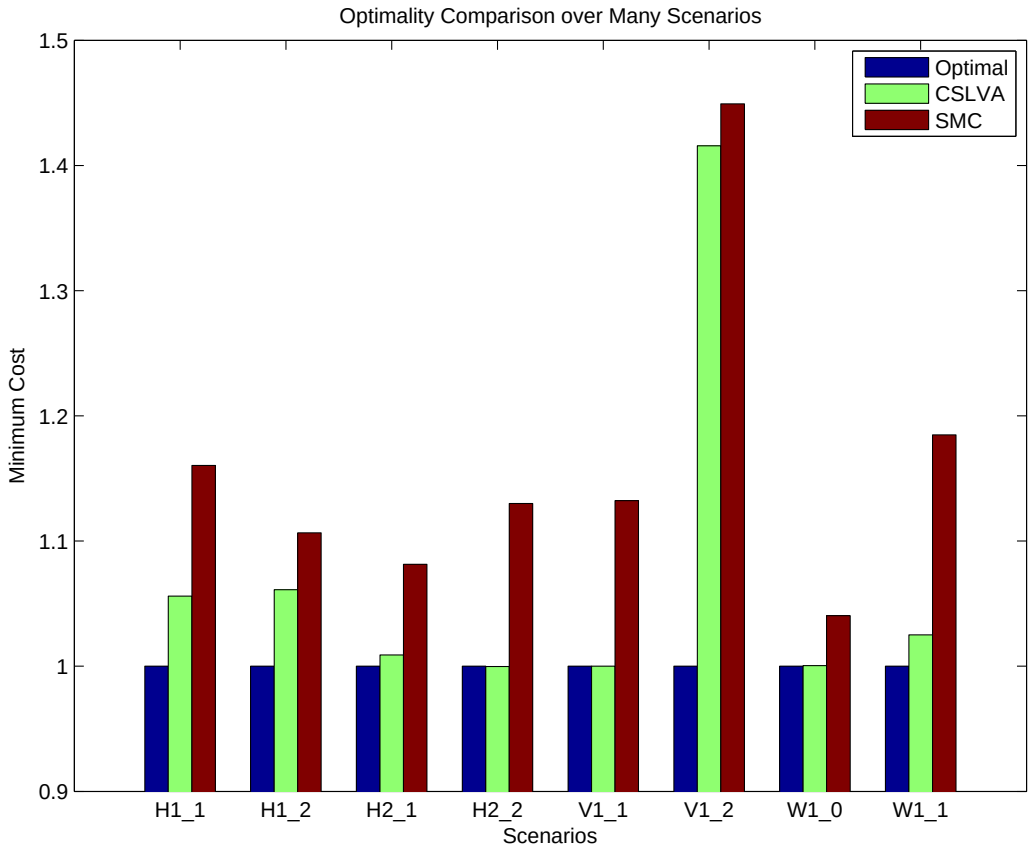

Figure 5.5: Optimality Comparison over eight scenarios. Optimal denotes the brute force exhaustive search method, CSLVA denotes the proposed constrained sequential List Viterbi-directed search method, and SMC denotes the sequential Monte Carlo-based search method.

Summary and Conclusions

A multiple model method is presented here for solving the aircraft conflict detection and resolution (CDR) problem in uncertainty. The method is applied to unmanned sense-and-avoid scenarios. In an effort to be as realistic as possible, the formulation and design revolve around a probabilistic foundation where safety is the number one concern followed by a good balance between swift response time and minimal fuel consumption. The method is broken up into two distinct parts, conflict detection and conflict resolution. For conflict detection, a randomized Monte Carlo algorithm is used that estimates the probability of conflict based on multiple model (MM) trajectory prediction. For conflict resolution, an algorithm is used that is based on the sequential list Viterbi algorithm wherein a list of maneuver sequences, ordered by non-decreasing cost, is sequentially built in order to find a lowest cost, conflict-free maneuver sequence.

Comprehensive simulation and performance evaluation show that the method detailed in this thesis: 1) improves the accuracy of the predicted probability of conflict, 2) optimizes the produced resolution trajectories, and 3) is computationally efficient. As previously shown, when the true predicted densities of an aircraft are in fact Gaussian mixtures, the probability of conflict can be more accurately estimated by using a multiple model framework. The optimality of the produced trajectory can be upheld by using a conflict resolution method that intelligently searches the solution set. Lastly, the number of PC computations and thus computation times can be greatly reduced by taking advantage of the problem's characteristics and adding efficiency improvements to the conflict resolution algorithm. The amount of computational time reduction is directly proportional to the difficulty of the scenario geometry. Although numerous other CDR methods exist already, it is fairly uncommon to find one that addresses the three aforementioned attributes all at the same time as successfully as the MM CDR method does.

Bibliography

[1] NextGen Implementation Plan 2013, June 2013.

[2] *Instrument Procedures Handbook*. U.S. Department of Transportation, 2015.

[3] H. Bai, D. Hsu, M. J. Kochenderfer, and W. S. Lee. Unmanned Aircraft Collision Avoidance using Continuous-State POMDPs. In *Robotics: Science and Systems*, Los Angeles, CA, 2011.

[4] Lars Blackmore, Masahiro Ono, Askar Bektassov, and Brian C Williams. A Probabilistic Particle-Control Approximation of Chance-Constrained Stochastic Predictive Control. *IEEE Trans. on Robotics*, 26(3):502–517, 2010.

[5] H. A. P. Blom and G. J. Bakker. Conflict Probability and Incrossing Probability in Air Traffic Management. In *Proc. Conference on Decision and Control*, Las Vegas, Nevada, U.S.A., Dec. 2002.

[6] R. Chamlou. Future Airborne Collision Avoidance – Design Principles, Analysis Plan and Algorithm Development. In *Proc. IEEE/AIAA 28th Digital Avionics Systems Conference*, pages 6.E.2.1–6.E.2.17, 2009.

[7] T Crescenzi, A Kaizer, T Young, J Holt, and S Biaz. Collision Avoidance in UAVs Using Dynamic Sparse A*. Technical report, Technical Report# CSSE11-02. Auburn University, 2011.

[8] Edsger W Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[9] A. Eele and J. Maciejowski. Comparison of Stochastic Methods for Control in Air Traffic Management. In *International Federation of Automatic Control World Congress*, September 2011.

[10] David Fish, Eric Westman, James Holt, and Saad Biaz. An Inverse Proportional Navigation Algorithm for Collision Avoidance among Multiple Unmanned Aerial Vehicles Technical Report #CSSE12-06.

[11] G. D. Forney. The Viterbi Algorithm. In *Proceedings of the IEEE*, volume 61, pages 268–278. IEEE, March 1973.

[12] Ben Gardiner, Wassen Ahmad, Travis Cooper, Matthew Haveard, James Holt, and Saad Biaz. Collision Avoidance Techniques for Unmanned Aerial Vehicles. Technical Report CSSE11-01, Auburn University, August 2011.

[13] Joel George and Debasish Ghose. A Reactive Inverse PN Algorithm for Collision Avoidance among Multiple Unmanned Aerial Vehicles. In *American Control Conference, 2009. ACC'09.*, pages 3890–3895. IEEE, 2009.

[14] I. Hwang and C. E. Seah. Intent-Based Probabilistic Conflict Detection for the Next Generation Air Transportation System. *IEEE Proceedings*, 96(12):2040–2059, 2008.

[15] V. P. Jilkov, J. H. Ledet, and X. R. Li. Improved Conflict Resolution Method for Unmanned Aircraft Sense-and-Avoid. In *Proc. 2016 International Conf. on Information Fusion*, Heidelberg, Germany, July 2016.

[16] V. P. Jilkov, X. R. Li, and J. H. Ledet. Improved Estimation of Conflict Probability for Aircraft Collision Avoidance. In *Proc. 2014 International Conf. on Information Fusion*, Salamanca, Spain, July 2014.

[17] V. P. Jilkov, X. R. Li, and J. H. Ledet. An Efficient Algorithm for Aircraft Conflict Detection and Resolution Using List Viterbi Algorithm. In *Proc. 2015 International Conf. on Information Fusion*, Washington, DC, July 2015.

[18] Nikolas Kantas, JM Maciejowski, and A Lecchini-Visintini. Sequential Monte Carlo for Model Predictive Control. In *Nonlinear Model Predictive Control*, pages 263–273. Springer, 2009.

[19] J. K. Kuchar and L. C. Yang. A Review of Conflict Detection and Resolution Modeling Methods. *IEEE Trans. on Intelligent Transportation Systems*, 1(4):179–189, 2000.

[20] X. R. Li and V. P. Jilkov. Survey of Maneuvering Target Tracking. Part I: Dynamic Models. *Aerospace and Electronic Systems*, 39:1333–1364, October 2003.

[21] X. R. Li and V. P. Jilkov. Survey of Maneuvering Target Tracking. Part V: Multiple-Model Methods. *Aerospace and Electronic Systems*, 41(4):1255–1321, Oct. 2005.

[22] W. Liu and I. Hwang. Probabilistic Aircraft Trajectory Prediction and Conflict Detection for Air Traffic Control. *AIAA Journal on Guidance, Control and Dynamics*, 34(6):1779–1789, 2011.

[23] Weiyi Liu and Inseok Hwang. Probabilistic Aircraft Midair Conflict Resolution Using Stochastic Optimal Control. *IEEE Trans. on Intelligent Transportation Systems*, 15(1):37–46, 2014.

[24] J. Maciejowski and A. Eele. Real-Time Optimisation-based PlanPlan and Scheduling of Vehicle Trajectories. In *2014 17th IEEE Mediterranean Electroltechnical Conference (MELECON)*, pages 305–309, Beirut, April 2014. IEEE.

[25] Anusha Mujumdar and Radhakant Padhi. Nonlinear Geometric and Differential Geometric Guidance of UAVs for Reactive Collision Avoidance. Technical report, DTIC Document, 2009.

[26] Christiane Nill and C-EW Sundberg. List and Soft Symbol Output Viterbi Algorithms: Extensions and Comparisons. *IEEE Trans. on Communications*, 43(2/3/4):277–287, 1995.

[27] R.A. Paielli and H. Erzberger. Conflict Probability Estimation for Free Flight. *Journal of Guidance, Control, and Dynamics*, 20(3):588–596, 1997.

[28] Jung-Woo Park, Hyon-Dong Oh, and Min-Jea Tahk. UAV Conflict Detection and Resolution Based on Geometric Approach. *International Journal of Aeronautical and Space Sciences*, 10(1):37–45, 2009.

[29] Joint Planning and Development Office. NextGen Avionics Roadmap, Version 2.0, September 30, 2011.

[30] M. Prandini, J. Hu, J. Lygeros, and S. Sastry. A Probabilistic Approach to Aircraft Conflict Detection. *IEEE Trans. on Intelligent Transportation Systems*, 1(4):199–220, 2000.

[31] M. Prandini and O. J. Watkins. Probabilistic Aircraft Conflict Detection. WP3 D3.2, IST-2001-32460 HYBRIDGE, May 2005.

[32] Arthur Richards and Jonathan P How. Aircraft Trajectory Planning with Collision Avoidance Using Mixed Integer Linear Programming. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3, pages 1936–1941. IEEE, 2002.

[33] Martin Roder and Raouf Hamzaoui. Fast Tree-Trellis List Viterbi Decoding. *IEEE Trans. on Communications*, 54(3):453–461, 2006.

[34] Jason Ruchti, Robert Senkbeil, James Carroll, Jared Dickinson, James Holt, and Saad Biaz. UAV Collision Avoidance Using Artificial Potential Fields Technical Report #CSSE11-03. Technical report, 2011.

[35] José J Ruz, Orlando Arévalo, Gonzalo Pajares, and Jesús M de la Cruz. Decision Making Among Alternative Routes for UAVs in Dynamic Environments. In *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, pages 997–1004. IEEE, 2007.

[36] Nambirajan Seshadri and C-EW Sundberg. List Viterbi Decoding Algorithms with Applications. *IEEE Trans. on Communications*, 42(2/3/4):313–323, 1994.

[37] Karin Sigurd and Jonathan How. UAV Trajectory Design Using Total Field Collision Avoidance. Technical report, Massachusetts Institute of Technology, 2003.

[38] S. Temizer, M. J. Kochenderfer, L. P. Kaelbling, T. Lozano-Perez, and J. K. Kuchar. Unmanned Aircraft Collision Avoidance using Partially-Observable Markov Decision Processes. Project Report ATC-356, MIT, Lincoln Laboratory, 2009.

[39] L. Yang, J. H. Yang, J. Kuchar, and E. Feron. A Real-time Monte Carlo Implementation for Computing Probability of Conflict. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Providence, RI, Aug. 2004.

[40] J. L. Yepes, I. Hwang, and M. Rotea. New Algorithms for Aircraft Intent Inference and Trajectory Prediction. *AIAA Journal of Guidance, Control, and Dynamics*, 30(2):370–382, 2007.

Vita

Jeffrey Ledet was born in Gretna, Louisiana in 1991 and has lived in New Orleans his whole life. He received his B.S. in Electrical Engineering with a minor in Computer Science from the University of New Orleans in May 2014. He started on his M.S. in Electrical Engineering in August 2014 and has been a research assistant in Dr. Li's and Dr. Jilkov's research group, Information and Systems Laboratory, since September 2013. He has co-authored three conference papers so far.