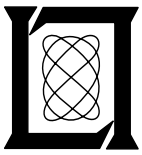
Project Report
ATC-371

Robust Airborne Collision Avoidance through Dynamic Programming

M.J. Kochenderfer
J.P. Chryssanthacopoulos

3 January 2011

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS



Prepared for the Federal Aviation Administration,
Washington, D.C. 20591

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
ATC-371		

4. Title and Subtitle	5. Report Date
Robust Airborne Collision Avoidance through Dynamic Programming	3 January 2010
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Mykel J. Kochenderfer and James P. Chryssanthacopoulos	ATC-371

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
MIT Lincoln Laboratory 244 Wood Street Lexington, MA 02420-9108	11. Contract or Grant No. FA8721-05-C-0002

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Department of Transportation Federal Aviation Administration 800 Independence Ave., S.W. Washington, DC 20591	Project Report
	14. Sponsoring Agency Code

15. Supplementary Notes

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology, under Air Force Contract FA8721-05-C-0002.

16. Abstract

The Traffic Alert and Collision Avoidance System (TCAS) uses an on-board beacon radar to monitor the local air traffic and logic to determine when to alert pilots to potential conflict. The current TCAS logic was the result of many years of development and involved the careful engineering of many heuristic rules specified in pseudocode. Unfortunately, due to the complexity of the logic, it is difficult to revise the pseudocode to accommodate the evolution of the airspace and the introduction of new technologies and procedures. This report summarizes recent advances in computational techniques for automatically deriving the optimal logic with respect to a probabilistic model and a set of performance metrics. Simulations demonstrate how this new approach results in logic that significantly outperforms TCAS according to the standard safety and operational performance metrics.

17. Key Words	18. Distribution Statement
	This document is available to the public through the National Technical Information Service, Springfield, VA 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	116	

FORM DOT F 1700.7 (8-72) Reproduction of completed page authorized

This page intentionally left blank.

EXECUTIVE SUMMARY

The Traffic Alert and Collision Avoidance System (TCAS), currently mandated on all large transport aircraft, has been shown to significantly reduce the risk of mid-air collision. TCAS uses an on-board radar to monitor the local air traffic and logic to determine when to alert pilots to potential conflict. If deemed necessary to prevent collision, TCAS will issue a resolution advisory to the pilots to climb or descend at a particular rate.

Developing robust collision avoidance logic that reliably prevents collision without excessive alerting is challenging due to sensor error and uncertainty in pilot response and, consequently, the future paths of the aircraft. The current TCAS logic was the result of many years of development and involved the careful engineering of many heuristic rules specified in pseudocode. Unfortunately, due to the complexity of the logic, it is difficult to revise to accommodate the evolution of the airspace and the introduction of new technologies and procedures.

An alternative to developing collision avoidance logic directly is to model the problem to be solved and then apply computational techniques to automatically derive the optimal logic with respect to a set of performance metrics. One advantage of this approach is that it is much easier to specify performance metrics and develop models of aircraft behavior than to develop robust pseudocode. The difficult task of ensuring that the logic performs well in all possible encounter situations is left to computers, which are better equipped than human designers to reason about the low-probability events that can occur during the course of an encounter that can result in collision.

The computational technique that enables the efficient optimization of the collision avoidance logic is called dynamic programming (DP). Although DP was originally developed over fifty years ago and has been applied to a wide variety of different problems, it has not been applied to collision avoidance until recently. In FY09, the TCAS Program Office sponsored an initial investigation into this method, and the research was summarized in Project Report ATC-360. While the results were promising, the performance was limited by assumptions made by the model.

This report discusses work in FY10 on relaxing previous assumptions. The new model allows advisories to be strengthened and reversed, as with the existing version of TCAS. Because it not possible to know exactly the future trajectories of aircraft, the provision for changing the initial advisory significantly reduces collision risk. Motion in three spatial dimensions is now captured by the model, allowing the logic to better determine when an alert is necessary. This report introduces probabilistic pilot response models, which can result in logic that is more robust to variability in response. The logic now uses explicit models to account for sensor noise, resulting in improved performance with realistic surveillance systems. This report shows how to coordinate maneuvers between aircraft and how to resolve threats from multiple aircraft.

The logic optimized by DP significantly outperforms TCAS in simulation according to the standard safety and operational performance metrics. Both the DP logic and the TCAS logic were evaluated on the same collection of encounters generated by a high-fidelity encounter model. Millions of simulated encounters show that the DP logic provides a lower probability of mid-air collision while reducing the alert rate. Although some further development is required, the DP approach appears viable for developing robust collision avoidance logic.

This page intentionally left blank.

ACKNOWLEDGMENTS

This report is the result of research and development sponsored by the TCAS Program Office at the FAA. The authors appreciate the assistance provided by the TCAS Program Manager, Neal Suchy, who has been supportive of pursuing new ideas and innovation for advancing aviation safety.

This work has greatly benefited from discussions with Thomas B. Billingsley, Barbara J. Chludzinski, Ann C. Drumm, Matthew W. M. Edwards, Leo P. Espindle, J. Daniel Griffith, William H. Harman, Tomás Lozano-Pérez, Leslie P. Kaelbling, James K. Kuchar, Donald E. Maki, Frans A. Oliehoek, Wesley A. Olson, David A. Spencer, Selim Temizer, Panayiotis C. Tzanos, Roland E. Weibel, Richard E. Williams, and M. Loren Wood.

This page intentionally left blank.

TABLE OF CONTENTS

This page intentionally left blank.

LIST OF ILLUSTRATIONS

LIST OF TABLES

This page intentionally left blank.

1. INTRODUCTION

The Traffic Alert and Collision Avoidance System (TCAS), currently mandated on all large transport aircraft, has been shown to significantly reduce the risk of mid-air collision. TCAS uses an on-board beacon radar to monitor the local air traffic and logic to determine when to alert pilots to potential near mid-air collision (NMAC). If deemed necessary to prevent collision, TCAS will issue a resolution advisory to the pilots to climb or descend at a particular rate.

The logic used to determine when to issue an alert and which advisory to issue was the result of decades of development. The logic is specified using pseudocode that has been validated through rigorous simulation studies to ensure that the system provides a high degree of safety while remaining operationally acceptable. Due to the complexity of the pseudocode, it is difficult to modify the logic. Much of the logic is likely to require re-engineering to accommodate new surveillance systems and next-generation air traffic control procedures.

This report explores a new approach to developing collision avoidance logic that has the potential to significantly improve safety while reducing the rate of unnecessary alerts. The approach involves leveraging recent algorithmic advances and modern computing power to optimize the logic based on probabilistic models of aircraft behavior and performance metrics. The probabilistic models can be modified to accommodate the anticipated evolution of the airspace, and the logic may be re-optimized as necessary with little development effort.

1.1 DESIGN CONSIDERATIONS

The following design considerations guided the development of the proposed approach.

- *Safety performance.* The primary objective of a collision avoidance system is to increase safety. Because flight tests can only test the system in relatively few encounter situations, much of the analysis required to prove the safety of the system must be done in simulation using an encounter model. During the development of TCAS, the NMAC rate was used as a safety performance metric. Historically, NMACs have been defined to occur when two aircraft come within 500 ft horizontally and 100 ft vertically. One of the primary goals of this research is to develop a system that has an NMAC rate lower than that of the existing version of TCAS. As will be discussed later, there are other ways to assess the safety of a collision avoidance system other than estimating NMAC rate.

- *Operational performance.* A collision avoidance system should not interfere with normal, safe flight operations. Hence, the system should avoid issuing unnecessary alerts. An excessive alert rate can result in decreased efficiency, collision with secondary aircraft, and pilot non-compliance. In order to reliably prevent collision, the system will need to alert in situations where it might later be found unnecessary. Because it is impossible to perfectly predict how an encounter will evolve, the system will need to act conservatively. Carefully balancing the operational considerations with safety considerations is an important part of developing a

collision avoidance system. A goal of this research is to develop a system that provides a lower alert rate than TCAS while enhancing safety.

- *Onboard computation.* TCAS currently makes decisions once per second, and a future system will likely be required to make decisions at least as frequently. Many of the algorithms suggested in the literature are not yet suitable for real-time use, but the method pursued in this report provides the required performance using current technology. The implementation of the proposed system is capable of making decisions in less than a millisecond on a single processor, which enables fast-time simulation of the system on millions of encounters during development.

- *Coordination.* If two TCAS-equipped aircraft encounter each other, they will coordinate their advisories over a low bandwidth communication channel. This coordination aims to prevent, for example, both aircraft from climbing into each other. A new system should also coordinate advisories to enhance safety. Although better coordination may be achieved through a higher bandwidth data link, this report assumes only the existing communication channel currently used by TCAS will be available for a new system.

- *Unequipped aircraft.* TCAS will avoid aircraft that are not TCAS-equipped, so long as they have a transponder. Although TCAS is not able to coordinate with non-TCAS aircraft, it still provides a significant safety benefit. Similarly, the new system must prevent collision even when the intruder is not equipped with a collision avoidance system.

- *Interoperability.* The introduction of a new collision avoidance system is likely to be gradual. Hence, any new system should be capable of interoperating with legacy versions of TCAS. Although the design proposed in this report allows interoperability, evaluating this will be the focus of future work.

- *Surveillance compatibility.* TCAS was designed for a particular surveillance system. Adjusting the existing logic to accommodate new surveillance systems, such as those that leverage global positioning system information, is nontrivial. Since surveillance technology is expected to evolve, the new collision avoidance logic should be sufficiently flexible to accommodate a wide variety of different surveillance systems.

- *Aircraft performance constraints.* The TCAS collision avoidance logic was intended primarily for large transport aircraft with certain capabilities, but the new system should be compatible with a wider variety of aircraft, including unmanned aircraft and those typically used for general aviation. Not only should the new system only issue advisories that are feasible for the aircraft on which it is installed, it should take into consideration the performance limits of other aircraft when coordinating advisories. For example, if an intruder is known to be unable to climb quickly, then TCAS may issue a descend advisory earlier than it would otherwise. Accommodating aircraft performance limitations is not a focus of this report, but it is a straightforward extension to the proposed method and will be pursued later.

- *Extensibility.* The airspace will evolve significantly over the next few decades with the introduction of next-generation air traffic control technologies. Any new collision avoidance system

will need to evolve with the airspace. Changing the TCAS pseudocode can be very difficult due to the complexity of the logic and the ways in which the various rules and parameters interact with each other. This report investigates an approach that allows the logic to be quickly re-optimized in response to changes in the airspace model.

- *Validation.* Due to its safety-critical nature, a collision avoidance system must be validated to ensure that the system provides the required level of safety with a high degree of confidence. The system proposed in this report can be validated using encounter models in Monte Carlo simulation, as has been done with TCAS. The nature of the approach taken in this report, however, permits additional, complementary validation techniques to be discussed later.

- *Verification.* It is important that a manufacturer's implementation of the logic be verified for correctness. TCAS implementations are currently verified against a set of test encounters. The system proposed in this report can also be verified against test encounters. Because the complexity of the logic can be encoded using numerical lookup tables that can be standardized and delivered to manufacturers, there is very little code for manufacturers to implement and hence less opportunity for errors to be introduced.

1.2 BACKGROUND

Developing a robust collision avoidance algorithm that reliably prevents collision without alerting excessively is challenging due to sensor error and uncertainty in the future behavior of the aircraft. A wide variety of different collision avoidance algorithms have been proposed, and they differ in how they predict the motion of aircraft. These methods may be categorized as follows:

- *Nominal trajectory propagation.* Many collision avoidance systems, including TCAS, propagate the most likely positions of the aircraft into the future. If an intruder is predicted to come within the NMAC volume or some enlarged conflict zone in the near future, the system decides that an alert is necessary. The system then predicts the effects of the available advisories and chooses the one that best resolves the NMAC. One problem with this approach is that it does not explicitly account for low-probability events that can lead to collision. To make the logic robust to deviations from the nominal predicted trajectories, algorithms that use this approach must rely upon complex heuristics. [1–4]

- *Worst-case trajectory propagation.* Other collision avoidance systems examine a range of possible maneuvers and determine whether any of them results in NMAC. One disadvantage of this approach is that it can result in an excessively high alert rate. Excessive alerting can result in NMACs with third-party aircraft, impaired efficiency, and decreased pilot compliance. [5–7]

- *Probabilistic trajectory propagation.* This method examines all possible future trajectories and accounts for their relative likelihood. Nominal and worst-case propagation are special cases of probabilistic propagation; nominal propagation assigns probability one to the most likely trajectory and worst-case propagation assigns equal probability to all trajectories. Accounting for the likelihood of different trajectories can result in robustness against low-probability hazardous events while maintaining an acceptable alert rate. [8–10]

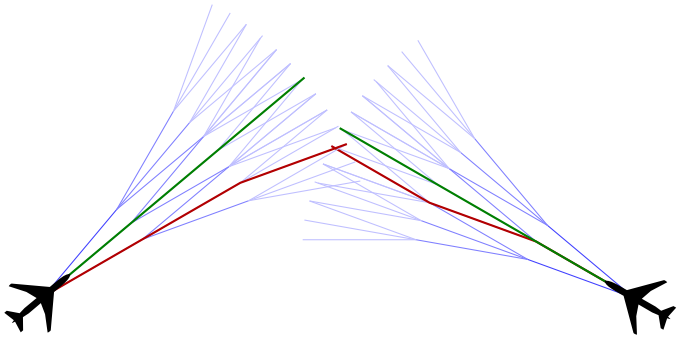


Figure 1. Trajectory propagation methods. Nominal trajectory propagation (green) only examines the most likely trajectory. Worst-case trajectory propagation (red) searches the space of possible trajectories for one that results in NMAC. Probabilistic trajectory propagation (blue) accounts for the relative likelihood of all trajectories.

These trajectory propagation methods are illustrated in Fig. 1. Surveys of these methods are found in [11, 12].

Although using a probabilistic model can result in significantly more robust collision avoidance, the majority of the algorithms in the literature use nominal trajectory propagation. With nominal trajectory propagation, only a single trajectory needs to be modeled, but with probabilistic trajectory propagation, all trajectories (or a large sample) need to be modeled. Hence, methods adopting nominal trajectory propagation typically require much less computation.

Several different probabilistic propagation methods have been proposed in the literature, including analytic, numerical approximation, Monte Carlo, and dynamic programming methods. The analytic and numerical approximation methods are fast, but they typically require strong assumptions to be made regarding the model of the aircraft dynamics. Monte Carlo methods are much more flexible, but many trajectory samples are typically required because NMACs are rare, although there has been research on ways to improve efficiency [13].

For the development of a future TCAS system, it appears that the best option is to use dynamic programming. As with Monte Carlo, dynamic programming can accommodate a wide variety of different models, but it requires relatively little computation during flight. Dynamic programming does all the intense computation offline during development and stores the results in a lookup table to be used later. Since dynamic programming requires the state space to be discretized, one potential limitation is that the number of state variables must be kept manageable to prevent the number of discrete states from exploding. However, this report shows that by carefully choosing the representation of the model, the number of discrete states can be kept manageable.

Collision avoidance algorithms that use probabilistic dynamic models differ in how they use the relative likelihood of different trajectories to make decisions about when to alert and which advisory to issue. Many of the existing algorithms convert the trajectory likelihoods into a probability of NMAC that is then compared against a threshold [14, 15]. Some algorithms alert the moment the probability of NMAC without evasive maneuvering exceeds some threshold, but there

are other strategies for delaying alerts. For example, one strategy is to delay alerting until a single advisory remains whose probability of NMAC is below some threshold.

An alternative to approaches based on NMAC probability thresholds is to define a cost metric for trajectories and optimize the collision avoidance logic to minimize the expected cost. Under certain assumptions to be discussed later, dynamic programming can be applied to compute the expected cost for each action (e.g., no alert, climb, or descend) at each state offline and store the results in a table. The table can then be used online to determine the best action from the current state. This report adopts a dynamic programming approach.

1.3 PROPOSED APPROACH

The proposed approach uses a probabilistic model of aircraft behavior and a cost metric to be optimized. In order to quickly decide what action is best given the history of sensor measurements, a significant amount of processing is done offline during the development phase. The results are summarized in large numerical tables that can be distributed to manufacturers and incorporated into their avionics. During flight, the collision avoidance system consults these tables in real time to decide which advisory to issue or whether to delay alerting.

1.3.1 Offline Development

The offline development of the system involves specifying a probabilistic dynamic model that captures the behavior of aircraft during close encounters and the response of pilots to resolution advisories. The dynamic model may be constructed based on a combination of expert judgment and recorded data. Because of the influence the dynamic model has on the optimized logic, it is critical that the model be thoroughly vetted with the development community.

A cost metric must also be specified that balances different safety and operational considerations, such as the alert rate and the NMAC rate. The optimal collision avoidance logic is defined to be the one that provides the lowest expected cost. The choice of cost function strongly influences the behavior of the logic. Optimizing the system according to different cost metrics and examining the resulting behavior in simulation may guide the choice of cost parameters, as will be discussed later in this report.

If the model and the cost metric meet certain criteria to be discussed later, dynamic programming may be applied to recursively compute the expected cost for each action from each state. These costs are what are stored in the numerical lookup tables to be used in real time to select actions. Computing these tables is relatively efficient because dynamic programming leverages the structure of the model and cost metric so that every possible future trajectory does not require explicit enumeration.

1.3.2 Real-Time Usage

At some update rate, the surveillance system will measure the positions of the intruder aircraft. The current TCAS surveillance system measures range and bearing of the local air traffic and

decodes the altitude information transmitted by their transponders. Other surveillance inputs, such as satellite-based position reports, may be used in the future, but this report focuses on the surveillance data currently provided by the TCAS surveillance system.

The current state of the world cannot be known exactly; it can only be inferred from the sensor measurements with some degree of confidence. The collision avoidance system updates a probability distribution over the set of possible states based on the sensor measurements. This probability distribution is used to compute the current expected costs for the various actions. The system simply executes the action with the lowest cost.

1.4 RECENT ADVANCES

The general approach pursued in this report was originally proposed in Project Report ATC-360 [12]. Although this report will review the important concepts from Project Report ATC-360, the focus will be on recent advances, which include:

- *Advisory changes.* The previous report assumed that once an advisory is issued, it cannot be changed. However, depending on how an encounter evolves, TCAS will either strengthen or reverse the original advisory. Because it is impossible to know exactly the future trajectories of aircraft, the provision for strengthening and reversing advisories is critical to reducing collision risk.
- *Motion in three spatial dimensions.* The model used to derive the logic in the previous report was essentially two dimensional. This report models aircraft motion in three spatial dimensions. Enhancing the realism of the model used to optimize the logic results in improved performance when deployed in the real world.
- *Probabilistic pilot response models.* The previous report assumed a deterministic pilot response to resolution advisories. The response model was based on that used by TCAS, which assumes that the pilot responds to an initial advisory in exactly 5 s with 1/4 g acceleration. Because pilots respond to their advisories with variable delay and strength, this report attempts to model this variability explicitly when optimizing the logic. Accounting for the relative likelihood of different pilot responses improves the robustness of the optimized logic.
- *State estimation.* The previous report assumed that the current state of the world, which includes the positions and velocities of the aircraft, is perfectly known. However, the TCAS sensor is imperfect, and so the state can only be estimated. This report uses a sensor model and dynamic model together to estimate a probability distribution over states based on sensor measurements. This distribution is then used to decide when to alert and which advisory to issue. Leveraging an explicit sensor model allows the logic to be robust against noisy measurements.
- *Coordination.* The previous report assumed that the intruder was not equipped with a collision avoidance system. If both aircraft are equipped with a collision avoidance system, they can coordinate their advisories over a communication channel to further enhance safety. This report explains how the logic can be optimized for coordination with other aircraft.

- *Multiple threat encounters.* The previous report assumed a single intruder, but as the density of the airspace increases, encounters with multiple intruder aircraft will become increasingly likely. This report explains how to accommodate encounter scenarios where multiple aircraft pose a collision threat.

1.5 OVERVIEW

Section 2 introduces Markov decision processes (MDPs) as a way to model collision avoidance problems when the aircraft are equipped with (nearly) perfect sensors and discusses one possible solution method. It discusses an extension to MDPs, called partially observable MDPs (POMDPs), that incorporates an observation model to account for state uncertainty. The section discusses the QMDP method for approximating the solution to a POMDP that leverages the solution of the underlying MDP.

Section 3 discusses how to model a simplified, two-dimensional collision avoidance problem as an MDP. In this simplified collision avoidance problem there is a single intruder who is not equipped with a collision avoidance system. Neither aircraft maneuvers horizontally. The own aircraft is equipped with perfect sensors and the pilot responds deterministically to advisories issued by the system. The optimal solution to the problem is obtained using dynamic programming (DP). The section presents an example encounter and various performance assessment results comparing the DP-derived logic to TCAS in head-on encounters.

Section 4 quantifies the robustness of the DP logic of Section 3 to modeling errors. It presents the results of evaluating the logic on a three-dimensional white-noise model as well as a higher-fidelity encounter model with a significantly different model structure. It introduces robust DP as a technique for making solutions less sensitive to particular choices of dynamic models. A state-based robustness analysis is performed to estimate the value of performance metrics starting from arbitrary states in the state space.

Section 5 extends the solution method of Section 3 to encounters in three dimensions and allows both aircraft to maneuver horizontally. To help mitigate the increased computational demands for a model of higher dimensionality, Section 5 introduces an approach to solving the problem when only some of the state variables are controllable. The approach involves decomposing the full problem into controlled and uncontrolled subproblems, both of which are solved separately using DP algorithms.

Section 6 discusses how to optimize the DP logic of Section 5 using probabilistic pilot response models in which the future response of the pilot to advisories is uncertain. The section presents two different pilot response models. Belief state filtering is used to update the belief regarding the compliance of the pilot to advisories.

Section 7 discusses how to optimize the logic in the presence of state uncertainty due to sensor noise. The own aircraft is modeled as being equipped with a TCAS-like sensor that receives measurements of intruder range, bearing, and altitude. The performance of the QMDP method is compared against TCAS in simulation. The results of a robustness analysis are also shown.

Section 8 discusses how to optimize the logic when the intruder is equipped with the same collision avoidance system as the own aircraft. The advisories issued by the systems must be coordinated so that the aircraft do not accidentally maneuver into each other. When the state is fully observable, the problem can be modeled as a multi-agent MDP, which can be solved efficiently using DP. With the introduction of sensor noise, the problem is transformed into a decentralized POMDP, which can be solved approximately using a number of strategies. Several coordination strategies are evaluated in simulation.

Section 9 generalizes the logic to handle encounters in which multiple intruder aircraft pose a threat to the own aircraft. Because adding new state variables for each intruder does not scale, this section examines different approximation methods, such as command arbitration and utility fusion, that decompose the full problem into smaller, more manageable subproblems. The section presents ways to visualize the behavior of the multithreat logic as well as simulation results.

Section 10 concludes the report, summarizing the major contributions, and outlines some directions for future work.

Appendix A describes the file format used to represent the expected cost table.

Appendix B overviews the Kalman filter and an extension to the Kalman filter, called the unscented Kalman filter, to approximate the effects of nonlinear dynamic and measurement functions. The section discusses the details of the horizontal and vertical filters used in Section 7.

2. PROBLEM FORMULATION

If a collision avoidance system is equipped with nearly perfect sensors, the problem of collision avoidance can be framed as a Markov decision process (MDP). In an MDP, the state of the world is assumed to evolve according to a fixed dynamic model. A solution to an MDP is a strategy that maximizes performance according to some metric. Due to the assumptions made by the dynamic model and the performance metric, to be outlined in this section, solutions may be found efficiently using a computational technique called dynamic programming (DP).

In cases where a collision avoidance system is equipped with noisy sensors, the problem is better framed as a partially observable Markov decision process (POMDP), which augments the MDP formulation with an observation model that is used to generate observations based on the state of the world. This section discusses both MDPs and POMDPs, and later sections will show how they can be applied to collision avoidance.

2.1 MARKOV DECISION PROCESSES

MDPs have been well studied since the 1950s and have been applied to a wide variety of problems [16–18]. They require that the dynamic model be Markovian, meaning that the probability of transitioning to state s' depends only on the current state s and action a. This probability is denoted $T(s, a, s')$, and T is often called the state-transition function. So long as sufficient information about the problem can be encoded in the state, the Markov assumption is usually valid. The set of possible states is denoted $\mathcal{S}$ and the set of possible actions is denoted $\mathcal{A}$.

Solving an MDP involves searching for a strategy for choosing actions, also called a policy, that maximizes a performance metric. Although policies can depend upon the entire history of states and actions, it is well known that under certain assumptions regarding the structure of the performance metric, it is sufficient to only consider policies that deterministically depend only on the current state without losing optimality [19]. Given a policy π, the action to execute from state s is denoted $\pi(s)$.

There are several common performance metrics, also called optimality criteria, typically used with MDPs. One common metric is the expected sum of instantaneous reward up to some fixed horizon. The optimal policy is the one that maximizes this metric. Because collision avoidance typically involves avoiding particular events, such as collision and alerting, it is a little easier to define the metric in terms of positive costs instead of negative rewards. The objective, then, is to minimize the expected sum of instantaneous costs. In this report, the word "cost" is used to mean "sum of instaneneous costs." When "instantaneous cost" is intended, it will always be written out as such.[1]

To make solving MDPs efficient, the immediate cost function is typically assumed to depend only on the current state and action. Although the immediate cost function may be probabilistic,

[1] In other texts, the expected sum of instantaneous costs is often called "cost-to-go."

this report assumes that the function is deterministic for simplicity. The cost of executing action a from state s is denoted $C(s, a)$.

One of the challenges when solving an MDP is deciding how to represent the policy. If the number of states is finite, then the mapping from states to actions can be represented using a table. For problems with continuous state variables, as is the case with collision avoidance, it is no longer possible to enumerate every possible state. The field of approximate dynamic programming has studied several different approaches [20], including the use of neural networks [21] and decision trees [22]. This report focuses on a representation that uses a grid-based discretization of the state space [23].

The first step in computing the optimal policy using a grid-based representation is to choose suitable cut points along the dimensions of the state space. The Cartesian product of the sets of cut points defines the vertices of the grid. These vertices correspond to the discrete states in the representation. One limitation of a grid-based method is that the number of discrete states grows exponentially with the number of state variables, making the storage and computation of the policy expensive. Hence, when designing an MDP to model a problem, it is important to include only the relevant state variables to help prevent the discrete state space from becoming too large.

Once a discretization scheme has been chosen, the original continuous transition function T must be redefined over the discrete state space. Given a start state and action to be executed, the resulting state distribution from the original transition function may include states that are not part of the discrete state space. Different strategies exist for arranging the resulting state distribution to have support only over the set of discrete states. The approach taken in this report is to sample from the resulting state distribution and apportion probability mass to the discrete states associated with the vertices of the cell enclosing the sample. Discrete states that are closer to the sample are assigned more weight [12, 23].

DP can be used to compute the expected cost from every discrete state when following an optimal K-step horizon policy π_K^*. The optimal expected cost function J_K can be used to determine the optimal policy. Computing J_K is done using an iterative process, starting with initializing the function $J_0(s) = 0$ for all states s. Given the function J_{k-1}, the function J_k is determined as follows:

$$J_k(s) = \min_a \left[C(s, a) + \sum_{s'} T(s, a, s') J_{k-1}(s') \right]. \tag{1}$$

This iteration is continued to the desired horizon. The state-action cost $J_K(s, a)$ with a K-step horizon is given by

$$J_K(s, a) = \min_a \left[C(s, a) + \sum_{s'} T(s, a, s') J_{K-1}(s') \right]. \tag{2}$$

An optimal K-step policy satisfies

$$\pi_K^*(s) = \arg\min_a J_K(s, a). \tag{3}$$

In general, there may be multiple actions that satisfy the equality above from a particular state. In this work, it is assumed that ties are broken according to some ordering over the set of available actions, resulting in a unique optimal policy. In the context of collision avoidance, it may make sense to give preference to less extreme advisories if they have the same expected cost as more extreme advisories.

Because the actions are categorical, one cannot simply interpolate the optimal policy directly to determine the optimal action from non-discrete states. Instead, one can interpolate the state-action function and from that determine the optimal action. Storing the optimal state-action function requires $|\mathcal{S}| \times |\mathcal{A}|$ entries, whereas storing the optimal policy requires only $|\mathcal{S}|$ entries. Because the number of actions available from a particular state is typically small in the collision avoidance domain, the storage of the state-action table can be compressed. The implementation used for the experiments in this report stores the state-action costs only for valid state-action pairs and uses an index file for fast lookups.

2.2 PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES

In POMDPs [24], the exact state of the world is not directly observable. Instead, observations are made that depend probabilistically on the current state of the world. The probability that observation o is made from state s is given by $O(s, o)$. If the observation space, denoted $\mathcal{O}$, is continuous, then $O(s, o)$ is a density. If $\mathcal{O}$ is finite, $O(s, o)$ is a probability mass.

In general, the initial state is unknown. The uncertainty in the initial state is represented by a distribution, often called a belief state. A belief state b assigns probability $b(s)$ to being in state s. The space of possible beliefs is denoted $\mathcal{B}$, which is a continuous space (each point in this space represents a probability distribution over $\mathcal{S}$) with dimensionality $|\mathcal{S}|$.

If the belief state is b, a is executed, and o is observed, the new belief state b' is given by

$$
\begin{aligned}
b'(s') &= \Pr(s' \mid o, a, b) && (4)\\
&\propto \Pr(o \mid s', a, b)\Pr(s' \mid a, b) && (5)\\
&= \Pr(o \mid s') \sum_s T(s, a, s')b(s) && (6)\\
&= O(s', o) \sum_s T(s, a, s')b(s). && (7)
\end{aligned}
$$

This process of updating the belief state follows directly from Bayes' rule and is known as state estimation or filtering [25].

In a POMDP, the belief state is used to determine which action to execute. Hence, policies are mappings from belief states to actions. As with MDPs, various optimality criteria can be defined for POMDPs, including the minimization of expected cost. Because the belief state space is high-dimensional and continuous, representing the optimal policy for a POMDP is much more difficult than for an MDP. Computing the optimal policy exactly, even for finite horizon problems, is typically infeasible [26]. However, many different approximation methods have been proposed in the literature [27–29].

2.3 FULL OBSERVABILITY APPROXIMATION

Several different POMDP solution strategies involve solving the problem assuming full observability using a DP algorithm [30–33]. If π_{MDP} is the policy assuming full observability, then one strategy called the "most likely state" (MLS) strategy [34] is as follows:

$$\pi(b) = \pi_{\text{MDP}}(\arg\max_s b(s)), \tag{8}$$

where b is the current belief state. The problem with the MLS strategy is that it does not take into account the possibility of being in different states. In the collision avoidance problem, for example, if the aircraft is slightly more likely to be in a "safe" state than an "unsafe" state, MLS will ignore the possibility of being in the unsafe state and recommend an overly optimistic action. The current version of TCAS uses point estimates of certain state quantities, although it incorporates some heuristics to assess the confidence in the estimates.

The POMDP approximation method adopted in this report involves choosing the optimal action under the assumption that the world will become fully observable at the next decision point. This method involves computing the state-action costs $J_{\text{MDP}}(s, a)$ assuming full observability. If the current belief state is b, then the action to be selected is given by

$$\pi(b) = \arg\min_a \sum_s b(s) J_{\text{MDP}}(s, a). \tag{9}$$

This heuristic, often called QMDP,[2] has been studied by others, and has been found to perform well on many problems but tends to fail in problems where taking particular actions results in a significant reduction in state uncertainty [30–33]. Because this method takes into consideration the full belief distribution when making decisions, it can provide more robust performance than MLS when state uncertainty is significant.

2.4 DISCUSSION

Before adopting an MDP framework for deriving collision avoidance logic, it is important to assess whether the assumptions made by the model are valid. One of the most significant assumptions it makes is that the dynamics are Markovian, which means that the current state must contain all the information relevant to predict the distribution of states at the next decision point. In some problems, it is not practical to add all the state variables necessary to make the dynamics Markovian because it makes the discretized state space intractably large. Depending on the problem, it may be possible to approximate a seemingly non-Markovian system with a simplified Markovian model and still obtain acceptable performance. One of the primary questions to be answered in this report is whether the collision avoidance problem can be adequately represented by an MDP that is simple enough to be solved in reasonable time while providing the desired safety and operational performance.

[2] In contexts where reward is to be maximized, as opposed to cost to be minimized, Q is often used in the literature [35] to represent the state-action value function. In general, $Q(s, a) = -J(s, a)$.

3. COLLISION AVOIDANCE IN TWO SPATIAL DIMENSIONS

The previous section introduced MDPs as a framework for modeling sequential decision problems and DP as a solution method. This section presents an MDP formulation to collision avoidance that makes the following assumptions:

- *Decisions at 1 Hz.* Decisions are made once per second. This is exactly the same frequency at which the current version of TCAS makes decisions.
- *No horizontal maneuvering.* The intruder is approaching head-on with a constant closure rate, and neither aircraft turns. Because the closure rate is assumed constant and there is no horizontal maneuvering during the encounter, the motion is essentially in two spatial dimensions: altitude and horizontal range. This assumption is relaxed in Section 5.
- *Deterministic pilot response.* It is assumed that the pilot responds deterministically to the resolution advisories issued by the system. In reality, there is significant variability in the response of pilots. Section 6 shows how to accommodate pilot response variability into the model.
- *Perfect sensors.* The collision avoidance system has perfect state information. In reality, the sensor system is likely to be noisy. This assumption is relaxed in Section 7.
- *Uncoordinated.* The intruder is not equipped with a collision avoidance system. If the intruder is also equipped with a collision avoidance system, it is important that their maneuvers be coordinated, as discussed in Section 8.
- *Single intruder.* There is a single intruder approaching the own aircraft. Although multithreat scenarios are relatively rare, this assumption is relaxed in Section 9.

Although this model is very simple, it serves as a base from which a more sophisticated collision avoidance system can be built, as discussed later in this report.

3.1 ACTION SPACE

The current version of TCAS issues advisories to the pilot through an aural annunciation, such as "climb, climb," and through a visual display. The visual display varies, but it is typically implemented on an instantaneous vertical speed indicator or a vertical speed tape or pitch cues on the primary flight display. The set of advisories issued by TCAS can be interpreted as target vertical rate ranges. If the current vertical rate is outside the target vertical rate range, the pilot should maneuver to come within the required range. If the current vertical rate is within the target range, a corrective maneuver is not required, but the pilot should be careful not to maneuver outside the range.

The current version of TCAS assumes that the pilot will respond to initial advisories with a $1/4\,\mathrm{g}$ maneuver to come within the target vertical range. Depending on how the encounter evolves,

TABLE 1

Reduced advisory set

	Vertical rate (ft/min)			
Name	Minimum	Maximum	Strength (g)	Available from
COC	$-\infty$	∞	0	All
DES1500	$-\infty$	-1500	1/4	COC
CL1500	1500	∞	1/4	COC
SDES1500	$-\infty$	-1500	1/3	CL1500, SCL1500, SCL2500, SDES2500
SCL1500	1500	∞	1/3	DES1500, SDES1500, SDES2500, SCL2500
SDES2500	$-\infty$	-2500	1/3	DES1500, SDES1500
SCL2500	2500	∞	1/3	CL1500, SCL1500

TCAS has the option to strengthen the original advisory or reverse the sense of the advisory. Once an advisory has been strengthened, TCAS may later weaken the advisory. All subsequent advisories are assumed to be responded to with a stronger 1/3 g maneuver.

The set of advisories used in this report for the DP logic is summarized in Table 1. In the table, COC stands for "clear of conflict," which means that no advisory has been issued or there is no longer a threat. The sense of the advisory is labeled either CL or DES, for either climb or descend, respectively. The prefix "S" indicates that a stronger response is assumed. Only the minimum and maximum rates are typically displayed to the pilot, and not the "strength" of the response, so the six advisories in the table (not including COC) only correspond to four different advisories to be displayed to the pilot. However, it is useful to distinguish the advisories according to the assumed strength of the maneuver when developing the MDP model.

Table 1 also indicates the availability of each advisory given the current advisory on display. For example, COC can be issued at any time. However, because DES1500 and CL1500 are initial advisories, they can only be issued if COC is on display to the pilot. The advisory SDES1500 can be issued following CL1500, SCL1500, and SCL2500, in which case it acts as a reversal, or following SDES2500, in which case it acts as a weakening. Because SDES1500 is a subsequent advisory, it cannot be issued following COC. It also cannot be issued following DES1500 because they are fundamentally the same advisory, differing only in strength.

The advisories in Table 1 are a subset of the advisories available in the current version of TCAS. Although it captures all of the "corrective" advisories issued by TCAS, it does not contain certain "preventive" advisories. TCAS can issue advisories such as "do not climb" to prevent an aircraft from climbing into another aircraft. These preventive advisories are less disruptive to the flight path of the aircraft. However, there is currently some debate within the TCAS development community about the utility of such advisories. Although this report does not incorporate preventive advisories, it is straightforward to include them. The computational and storage requirements of the MDP approach scale linearly with the addition of new actions.

The collision avoidance system may choose to initially issue either DES1500 or CL1500. Following an initial 5 s delay, the pilot of the own aircraft will respond with a 1/4 g maneuver,

if necessary, to achieve the required vertical rate. Following the initial advisory, the system may choose to either terminate the advisory, strengthen the advisory, or reverse the advisory. If the advisory is strengthened, the pilot will maneuver at 1/3 g after a 3 s delay to achieve a vertical rate of 2500 ft/min in the direction of the original advisory. If the advisory is reversed, the pilot will maneuver at 1/3 g after a 3 s delay to achieve a vertical rate of 1500 ft/min in the opposite direction of the original advisory [36].

3.2 STATE SPACE

The state of the world is represented using five variables:

- h: altitude of the intruder relative to the own aircraft,
- $\dot{h}_0$: vertical rate of the own aircraft,
- $\dot{h}_1$: vertical rate of the intruder aircraft,
- τ: time to closest approach (horizontally), and
- s_{RA}: the state of the resolution advisory.

The variable τ is the time to horizontal closest approach, which is the same as the time to zero horizontal separation in this model because the encounter is assumed head-on with no turning. Instead of τ, the horizontal range could have been used, but τ allows the MDP to be formulated independently of the actual closure rate, assumed to be constant.

The variable s_{RA} is discrete. It allows the system to track which advisory is currently active (if any) and the time remaining until the pilot responds. For the advisories in Section 3.1, s_{RA} can assume one of 23 different values. The state COC indicates that there is no resolution advisory currently active. If DES1500 is issued, for example, the state transitions to DES1500-4, indicating that there are four more seconds until the pilot will execute the descend. If the descend advisory is continued, the state will countdown deterministically, as in Fig. 2. The pilot begins executing the descent at DES1500-0. The transition model for CL1500 is similar.

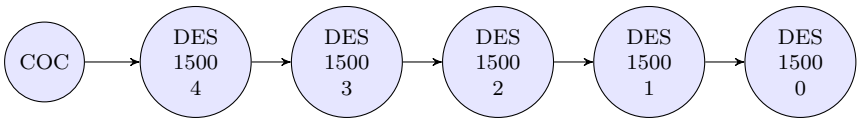


Figure 2. Advisory transition model when DES1500 is issued.

Fig. 3 shows the transition model when a subsequent advisory, SDES1500, is issued. The SDES1500 advisory can be issued following any climb advisory or SDES2500. It cannot be issued following COC or DES1500. When SDES1500 is issued for the first time, the state transitions to SDES1500-2, indicating two seconds until execution. If SDES1500 is continued, the state counts

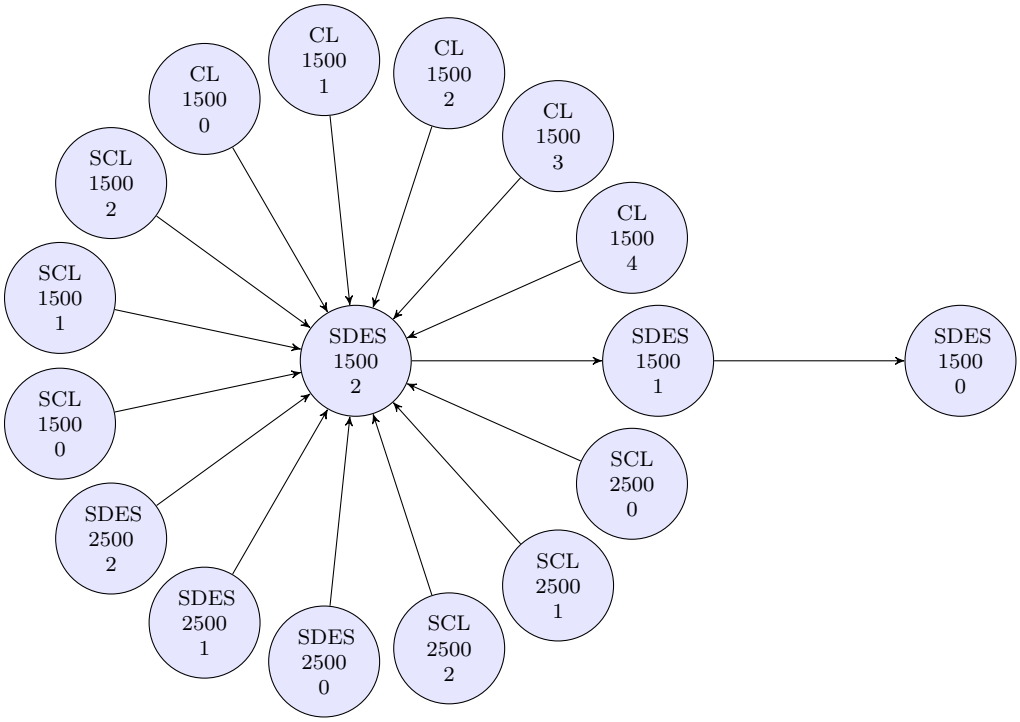


Figure 3. Advisory transition model when SDES1500 is issued.

down until reaching SDES1500-0, at which point the advisory is followed. The transition models for SCL1500, SDES2500, and SCL2500 are similar. When the advisory is terminated, the state transitions to COC immediately, discontinuing any response of the pilot to advisories.

The state space is discretized using a multidimensional grid. This report uses the discretization shown in Table 2, which results in 8.7 million grid vertices that correspond to discrete states. The discretization can be made finer to improve the quality of the discrete model approximation, but it would be at the expense of additional computation and storage to calculate and store the expected cost table.

TABLE 2

Discretization scheme

Variable	Grid Edges
h	$-1000, -900, \ldots, 1000\,\text{ft}$
$\dot{h}_0$	$-2500, -2250, \ldots, 2500\,\text{ft/min}$
$\dot{h}_1$	$-2500, -2250, \ldots, 2500\,\text{ft/min}$
τ	$0, 1, \ldots, 40\,\text{s}$
s_{RA}	N/A (already discrete)

3.3 DYNAMIC MODEL

The dynamics of the aircraft involved in the encounter are governed by sequences of accelerations. These accelerations are used to update the vertical rates of the aircraft and, consequently, their positions. The maximum vertical rate of both aircraft is assumed to be $\pm 2500\,\text{ft/min}$, although this is easily changed at the expense of a larger state space. In a real system, the limits can be adjusted to meet the performance constraints for the particular aircraft.

When pilots are not following a resolution advisory due to the response delay or the absence of an advisory, the aircraft follow a white-noise acceleration model. At each step, the aircraft selects an acceleration from a zero-mean Gaussian distribution with $\sigma_{\ddot{h}}$ standard deviation. If the $\sigma_{\ddot{h}}$ parameter is small, the aircraft tend to fly in straight lines. Large values for $\sigma_{\ddot{h}}$ can lead to significant variability in the paths of the aircraft. This section assumes $\sigma_{\ddot{h}} = 3\,\text{ft/s}^2$, but this parameter may be estimated from radar data. Following the response delay to an advisory, it is assumed that the pilot maneuvers with exactly the prescribed acceleration to come within the target vertical rate range. Once within the target vertical rate range, the aircraft resumes white-noise vertical accelerations.

In order to apply DP, the dynamic model must be discretized into a discrete transition model $T(s, a, s')$. One way to do this is to define $\Pr(s \mid \mathbf{x})$, a probability distribution over discrete states given a continuous state. Several different schemes can be used; this report uses multilinear interpolation because it was shown to work well for the collision avoidance problem in ATC-360. In this scheme, probability is assigned to the discrete states associated with the vertices of the grid cell that encloses the continuous state $\mathbf{x}$. The probability assigned to a state s is based on how close it is to $\mathbf{x}$. If the probabilities are treated as weights, the weighted average of the vertices enclosing $\mathbf{x}$ is exactly $\mathbf{x}$. Once $\Pr(s \mid \mathbf{x})$ is specified using multilinear interpolation or some other method, then the discrete transition model can be computed as follows:[3]

$$T(s, a, s') = \Pr(s' \mid s, a) \tag{10}$$

$$= \sum_{\mathbf{x}'} \Pr(s' \mid \mathbf{x}') \Pr(\mathbf{x}' \mid s, a). \tag{11}$$

Given a state and action, $\Pr(\mathbf{x}' \mid s, a)$ can be nonzero for a nonfinite set of continuous next states. To build the discrete transition function, this report relies upon sigma-point sampling. Other sampling methods may be used, but sigma-point sampling has been shown to work well for the collision avoidance problem in ATC-360, and it has the desirable property of being deterministic. The only noise in the dynamic model is due to sampling $\ddot{h}_0$ and $\ddot{h}_1$, the accelerations of the aircraft when an advisory is not being followed. The following acceleration pairs are deterministically generated using sigma-point sampling: $(0, 0), (\sigma_{\ddot{h}}, 0), (-\sigma_{\ddot{h}}, 0), (0, \sigma_{\ddot{h}}), (0, -\sigma_{\ddot{h}})$. Each sample is assigned weight $1/6$, except for the first one, which is assigned weight $1/3$. When executing action a from state s with acceleration pair $(\ddot{h}_0, \ddot{h}_1)$, the resulting state is uniquely given by the model

$$\mathbf{x}' = f(s, a, \ddot{h}_0, \ddot{h}_1). \tag{12}$$

[3] In reality, $\Pr(\mathbf{x}' \mid s, a)$ is a density, but the probability density function is approximated by a probability mass function through sampling.

TABLE 3

Event costs

NMAC	Alert	Strengthening	Reversal	Clear of Conflict
1	0.01	0.009	0.01	-1×10^{-4}

Given N samples of the accelerations (five in the sigma-point scheme just described), $\Pr(\mathbf{x}' \mid s, a)$ is approximated by

$$\Pr(\mathbf{x}' \mid s, a) = \sum_{n=1}^{N} \delta(\mathbf{x}', f(s, a, \ddot{h}_0^{(n)}, \ddot{h}_1^{(n)}))\Pr(\ddot{h}_0^{(n)}, \ddot{h}_1^{(n)}), \tag{13}$$

where $\delta(X, Y) = 1$ if $X = Y$ and 0 otherwise. The probability $\Pr(\ddot{h}_0^{(n)}, \ddot{h}_1^{(n)})$ is simply the weight assigned to the sample. Equation (13) may be combined with Eq. (11) to define the discrete transition probabilities.

Computing the distribution over discrete next states scales $O(N2^D)$, where N is the number of samples and D is the number of dimensions in the state space. Both N and D are 5 for the model in this section, so the required computation is quite modest. If the dimensionality of the state space is required to grow significantly larger to accommodate additional state variables not currently modeled, simplex interpolation may be a better option than multilinear interpolation because its computation scales linearly instead of exponentially with the number of variables [22, 23, 37].

3.4 COST FUNCTION

The costs of various events are summarized in Table 3. NMAC, in this section, occurs when $|h| < 100\,\text{ft}$ and $\tau = 0\,\text{s}$. The 100 ft threshold was chosen because it corresponds to the vertical separation of an NMAC used historically as a metric for evaluating TCAS. The costs were chosen somewhat arbitrarily, but the cost of alerting was made low compared to the cost of NMAC. The cost of strengthening was made lower than that of reversing because the required maneuvering would be less severe. A small negative cost, called the clear-of-conflict reward, is awarded at every time step the system is not alerting to provide some incentive to discontinue alerting after the encounter has been resolved.

This report focuses on a relatively small number of cost factors, but additional factors can be incorporated to address other safety or operational considerations. It may be desirable to add another state variable that represents the altitude above ground and then add a large penalty for issuing down-sense advisories at low altitude to prevent collision with terrain. Operationally, it may be desirable to discourage issuing advisories when aircraft are flying level and separated by more than 500 ft in altitude. If the state space does not require expanding, the computation required to construct the expected cost table grows linearly with the number of cost factors and the storage required remains constant. Online execution of the logic also remains constant.

3.5 OPTIMAL POLICY

DP, as introduced in Section 2, can be applied directly to the discretized model to obtain a discrete expected cost table $J(s, a)$. The discretization in Table 2 leads to an expected cost table of 263 MB using the sparse action representation described in Appendix A. Computing the expected cost table requires less than a minute on a single 3 GHz Intel Xeon core.

Since the collision avoidance logic is critical to safety, it is important for humans to understand and be able to anticipate the behavior of the system. Because the logic makes decisions based on values in an expected cost table, which is not directly informative to a human, it is necessary to develop ways to visualize the logic. Visualization is also important in building confidence that the logic produced through computer optimization is sensible.

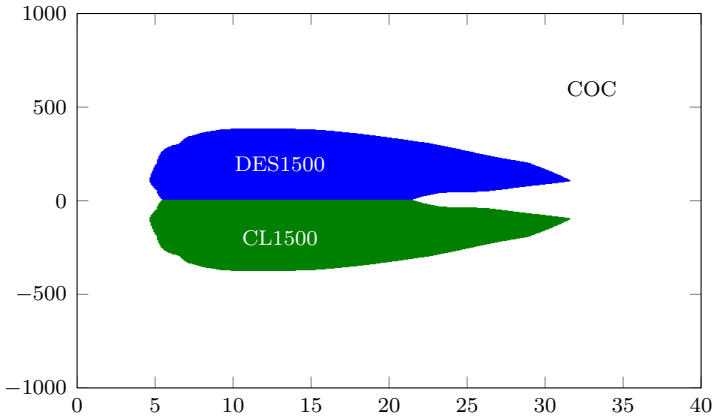
Figure 4 shows plots that indicate the optimal action for different slices of the state space. In the first plot, both aircraft are initially level and no advisory has been issued. The blue region indicates where the logic will issue a descend advisory, and the green region indicates where the logic will issue a climb advisory. The colored region (either green or blue in this first plot) is called the alerting region.

There are two notable properties of the shape of the alerting region. The first is that no advisory is issued when $\tau \leq 5\,\text{s}$, which is due to the 5 s pilot response delay. Alerting less than 5 s prior to collision makes no difference in preventing collision, so the optimized logic chooses not to accrue the cost of alerting. In reality, there is likely to be some nonzero chance that the pilot will respond sooner. If the model is adjusted to allow some probability of the pilot responding in fewer than five seconds, then the alerting region will extend further to the left. This will be discussed in Section 6.

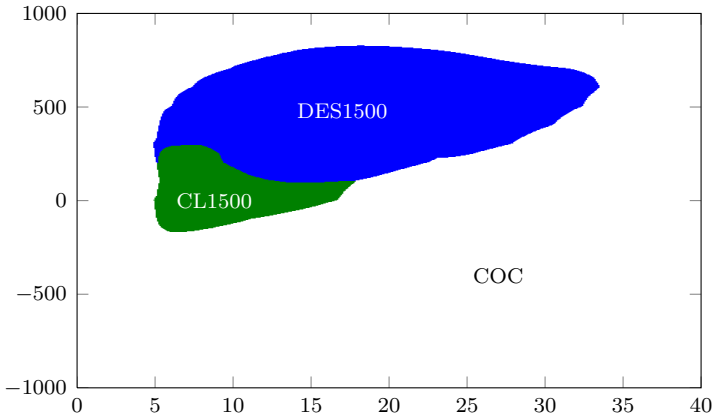
The second notable property of the first plot is the notch on the right side of the alerting region. In cases where the intruder is close to co-altitude with the own aircraft, the logic will delay alerting. It waits until it is more certain whether the intruder will end up above or below the own aircraft before committing to an advisory.

In the second plot, the own aircraft is initially climbing at 1500 ft/min while the intruder is initially level. The shape of the alerting region is more complex and lacks the symmetry of the first plot due to relative motion in the vertical plane. One notable property of this plot is that, in some situations, the system issues a climb when the intruder is above. In these situations, because the own aircraft is already climbing, there is insufficient time for the own aircraft to accelerate downwards to pass below the intruder.

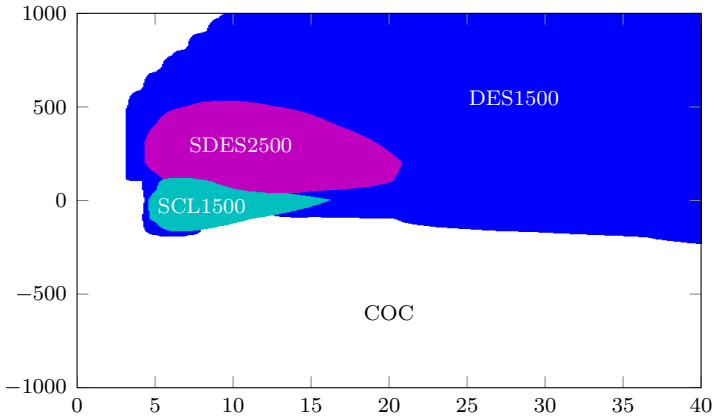
The third plot shows the alerting region when the own aircraft is climbing at 1500 ft/min and the intruder is level, as in the second plot, but an initial descend advisory was issued two seconds previously. Unless the advisory is canceled, strengthened, or reversed, the pilot will begin descending in three seconds. The white region indicates where the system will discontinue the advisory, and the blue region shows where the system will continue with the advisory that has already been issued. In some situations, the logic will strengthen the advisory to a descend at 2500 ft/min or reverse the advisory to a climb.



(a) $\dot{h}_0 = 0\,\text{ft/min}, \dot{h}_1 = 0\,\text{ft/min}, s_{\text{RA}} = \text{COC}$



(b) $\dot{h}_0 = 1500\,\text{ft/min}, \dot{h}_1 = 0\,\text{ft/min}, s_{\text{RA}} = \text{COC}$



(c) $\dot{h}_0 = 1500\,\text{ft/min}, \dot{h}_1 = 0\,\text{ft/min}, s_{\text{RA}} = \text{DES1500-3}$

Figure 4. Optimal action plots. Horizontal axis indicates τ (s) and the vertical axis indicates h (ft).

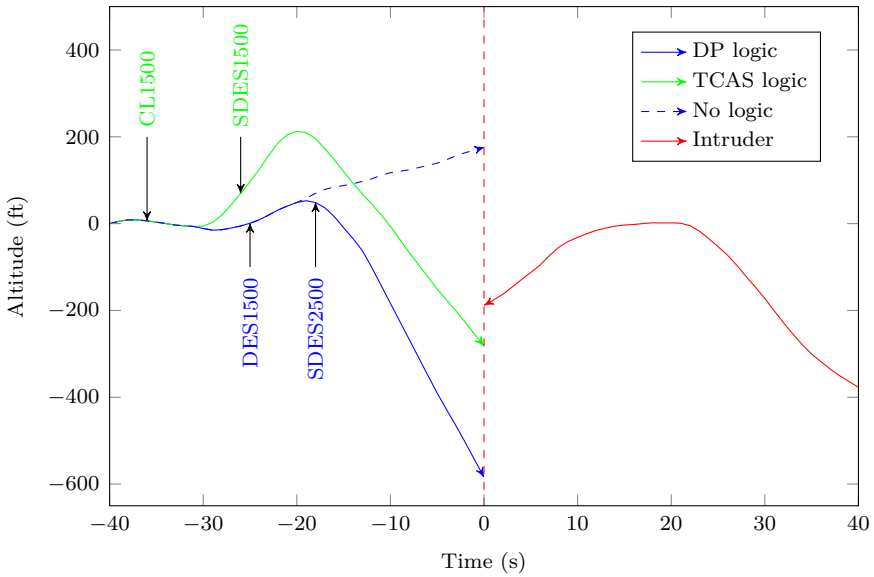


Figure 5. Example encounter comparing the performance of the DP logic against the current TCAS logic. The own aircraft approaches from the left, and the intruder approaches from the right.

3.6 EXAMPLE ENCOUNTER

Figure 5 shows an example head-on encounter comparing the behavior of the DP logic and the current TCAS logic, Version 7.1. The DP logic, 15 s into the encounter, issues a descend advisory to pass below the intruder as the intruder is climbing at approximately 1100 ft/min. The expected cost for issuing a descend advisory at this time is 0.012, smaller than the expected costs for not issuing an advisory (0.013) or issuing a climb advisory (0.020). The pilot begins descending 5 s later. As the intruder levels off later in the encounter, the DP logic strengthens the descend advisory at 23 s. The pilot begins to strengthen the descent 3 s later. Once safely separated in altitude, the DP logic issues a COC advisory. The vertical separation when $\tau = 0$ is 396 ft.

The performance of TCAS was evaluated using a constant 500 ft/s horizontal closure rate and an initial altitude of 43,000 ft, which puts TCAS into its highest sensitivity level [38]. When the own aircraft is 324 ft above the intruder 4 s into the encounter, TCAS issues a climb advisory because it anticipates, using straight-line projection, that by climbing it can safely pass above the intruder. Ten seconds later, when the own aircraft is only 141 ft above the intruder, TCAS reverses the climb to a descend because it projects that maintaining the climb will not provide the required level of separation. As the intruder begins to level off and then descend, the descend advisory becomes ineffective. The vertical separation when $\tau = 0$ is 94 ft.

3.7 PERFORMANCE ASSESSMENT

The overall performance of the system can be estimated by generating a large collection of encounters and running them in simulation. The results in this section use the (continuous) dynamic

TABLE 4

Performance evaluation on head-on encounters

Metric	DP Logic	TCAS Logic
NMACs	3	169
Alerts	690,406	994,317
Strengthenings	92,946	40,470
Reversals	9569	197,315

model assumed by the MDP. The simulation of TCAS (Version 7.1) used a constant 500 ft/s horizontal closure rate. To generate the initial state of the encounter, $\dot{h}_0$ and $\dot{h}_1$ are chosen from a uniform distribution from -1000 ft/min to 1000 ft/min. The variable τ is set to 40 s and s_{RA} is set to COC. So that a significant fraction of encounters ($\sim$13%) have a vertical miss distance of less than 100 ft, h is set to $\tau(\dot{h}_0 - \dot{h}_1) + \eta$, where η is chosen from a zero-mean Gaussian with 25 ft standard deviation.

Table 4 summarizes the results of evaluating the DP and TCAS logics on one million encounters. As the table shows, TCAS results in 50 times more NMACs than the DP logic, even though it alerts 45% more frequently. Although TCAS strenghtens half as frequently as the DP logic, it reverses 20 times more frequently. It should be emphasized, however, that the simple model was designed to stress test the logic, not provide accurate estimates of performance in the current airspace. High-fidelity encounter models based on recorded surveillance data have been developed for this purpose and will be discussed in the next section [39].

3.8 SAFETY CURVE

A safety curve is one way to visualize the effect of varying a system parameter on safety and alert rate. Figure 6 shows the safety curves for the DP logic and TCAS. The DP logic safety curve was produced by varying the cost of alerting from zero to one. The other event costs were held fixed. The upper-right part of the curve corresponds to costs of alerting near zero and the lower-left part corresponds to costs near one.

The sensitivity of TCAS, in terms of alerting thresholds and other parameters, varies discretely with altitude. Above 42,000 ft, TCAS is at its highest sensitivity level and will maneuver earlier and more aggressively to prevent collision. The safety curve for TCAS was generated by varying the initial altitude. The upper-right part of the curve corresponds to greater altitudes.

The safety curve shows that the DP logic can exceed or meet the level of safety provided by TCAS while alerting far less frequently. Plotting safety curves like the one in Fig. 6 can aid in choosing the relative cost for alerting. The alert cost can be chosen to provide the lowest rate of alerting for a given required safety level. Alternatively, one can choose the maximum acceptable alert rate and use the appropriate alert cost to maximize safety.

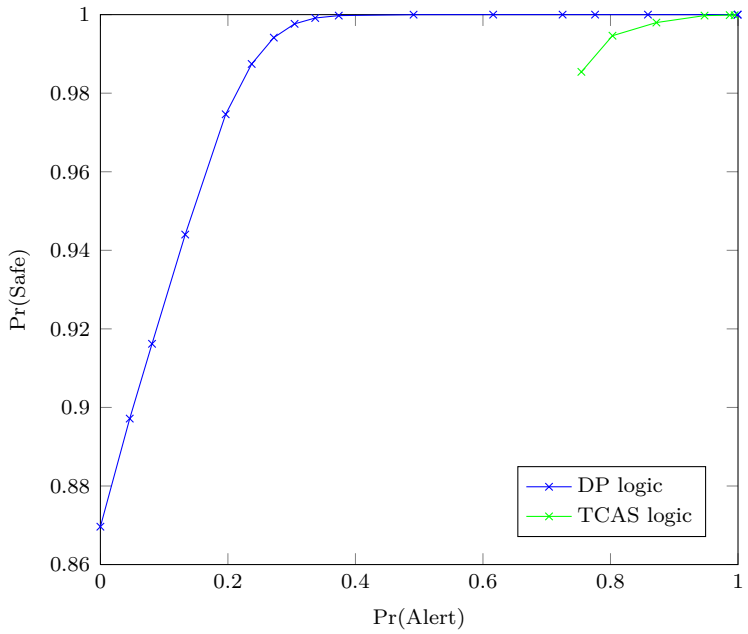


Figure 6. Safety curve. Each point on the curve was estimated from 100,000 simulations.

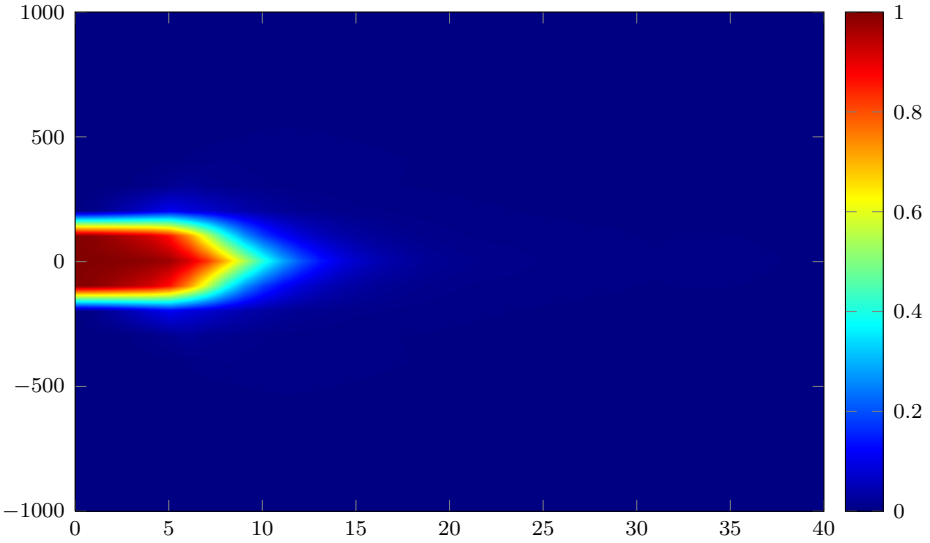
3.9 EFFICIENTLY EVALUATING PERFORMANCE METRICS

A DP algorithm known as iterative policy evaluation can efficiently evaluate the logic on metrics that are different from those used to optimize the logic [35]. During the process of validating the logic, it may be useful to determine where in the state space the system has difficulty preventing NMAC. It may also be useful to determine the probability that the system will eventually alert from different starting states. Figure 7 shows these metrics evaluated on a slice of the state space.

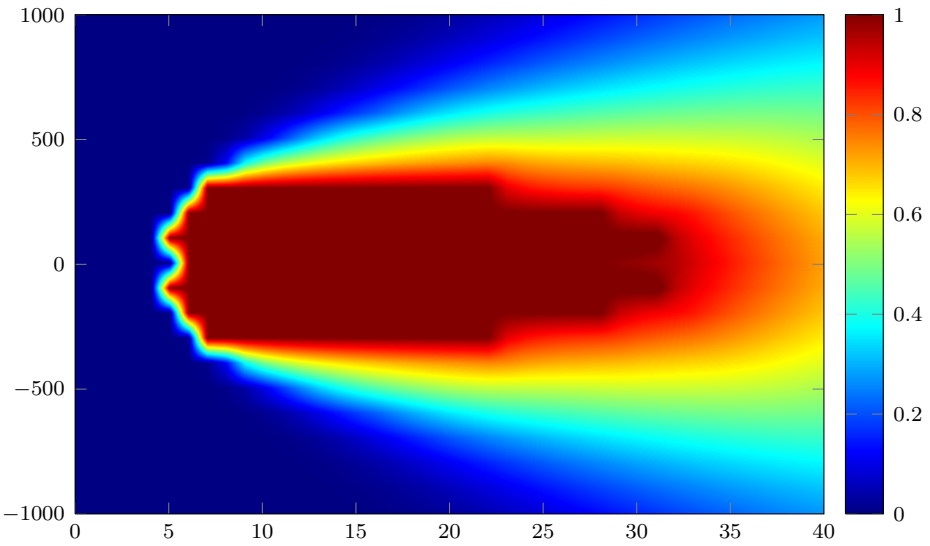
Prior TCAS safety analyses used Monte Carlo simulation to evaluate performance. Running an adequate number of Monte Carlo simulations from each discrete state would be prohibitively expensive. For example, even if Monte Carlo required only one second to compute the probability of NMAC from each state, it would require 100 days of continuous computation to estimate the probability of NMAC from all 8.7 million states. DP can do this computation in under a minute.

3.10 DISCUSSION

This section showed that, compared to the current TCAS logic in head-on encounters with fixed closure rates, the optimized logic provides greater safety while alerting much less frequently. Although the model only represents motion in two spatial dimensions, the next section will demonstrate how well it does in three-dimensional encounters, even when the model used for optimization does not match the one used for evaluation. Performance in three dimensions, however, can be improved even further by explicitly modeling motion in the horizontal plane. This enhancement to the logic is presented in Section 5.



(a) Probability of NMAC.



(b) Probability of alert.

Figure 7. Metric evaluation on the state slice where $\dot{h}_0 = 0\,\text{ft/min}, \dot{h}_1 = 0\,\text{ft/min}, s_{RA} = COC$. *Horizontal axis indicates* τ *(s) and the vertical axis indicates* h *(ft).*

4. ROBUSTNESS ANALYSIS

The previous section presented a way for computing the optimal collision avoidance strategy assuming a simple, head-on model. It presented performance results from simulations that used the same dynamic model that was used for optimization. Because it is unlikely that the model used for optimizing the logic will exactly match the real world, it is important to quantify the robustness of the logic to modeling errors. This section analyzes how the performance of the logic, in terms of safety and operational metrics, degrades as the model used for evaluating the logic diverges from the model used for optimizing the logic.

This section uses the same simple model and cost function from the previous section to optimize the logic. When evaluating the logic in three spatial dimensions,

$$\tau = \begin{cases} -r/\dot{r} & \text{if } \dot{r} < 0, \\ \infty & \text{otherwise,} \end{cases} \tag{14}$$

where r is the horizontal range to the intruder and $\dot{r}$ is the horizontal closure rate. The next section will present a more principled method for handling three-dimensional dynamics, but this simple method will be used to provide a rough estimate of the robustness of the approach.

This section describes two encounter models for evaluating the robustness of the optimal logic to modeling errors. These models are described in the next two sections.

4.1 WHITE-NOISE ENCOUNTER MODEL

The white-noise encounter model uses the dynamic model of Section 3.3 to model the vertical dynamics. The process noise parameter $\sigma_{\ddot{h}}$ controls the amount of vertical accelerating the aircraft do. The horizontal dynamics are also modeled by a white-noise acceleration model. Each aircraft moves in the horizontal plane in response to independent random accelerations generated from a zero-mean Gaussian with a standard deviation of $3\,\mathrm{ft/s^2}$.

Encounters are initialized by randomly generating the initial ground speeds of both aircraft, s_0 and s_1, from a uniform distribution from $100\,\mathrm{ft/s}$ to $500\,\mathrm{ft/s}$. The horizontal range between the aircraft is set to

$$r = t_{\text{target}}(s_0 + s_1) + u_r, \tag{15}$$

where u_r is selected from a zero-mean Gaussian with $500\,\mathrm{ft}$ standard deviation. The parameter t_{target}, set to $40\,\mathrm{s}$ in the experiments, controls the expected time to NMAC.

The bearing of the intruder aircraft with respect to the own aircraft is sampled from a zero-mean Gaussian distribution with a $2°$ standard deviation. The heading of the intruder relative to the heading of the own aircraft is sampled from a Gaussian distribution with a mean of $180°$ and a standard deviation of $2°$.

The initial vertical rates, $\dot{h}_0$ and $\dot{h}_1$, are drawn from a uniform distribution ranging from $\pm 1000\,\text{ft/min}$. The altitude of the own aircraft h_0 is initialized to 43,000 ft. The initial altitude of the intruder is

$$h_0 + t_{\text{target}}(\dot{h}_0 - \dot{h}_1) + u_h, \tag{16}$$

where u_h is selected from a zero-mean Gaussian with 25 ft standard deviation.

4.2 CORRELATED ENCOUNTER MODEL

The white-noise encounter model is useful for evaluating the robustness of the logic as the process noise parameter $\sigma_{\ddot{h}}$ is varied. However, the assumption that the vertical motion is governed by a white-noise acceleration model is the same as that assumed in the logic optimization process. To determine the robustness of the logic to a significantly different model structure, the performance of the logic is evaluated on a higher-fidelity encounter model, called the correlated encounter model, derived from nine months of recorded radar data [39, 40].

The correlated encounter model represents the variables governing the initialization of the encounter as a Bayesian network [41]. The Bayesian network represents a probability distribution over sixteen variables such as airspace class, category of aircraft, initial airspeed, and acceleration. The dynamics of the aircraft are represented by a dynamic Bayesian network, which captures the changes in turn rate and vertical rate over time.

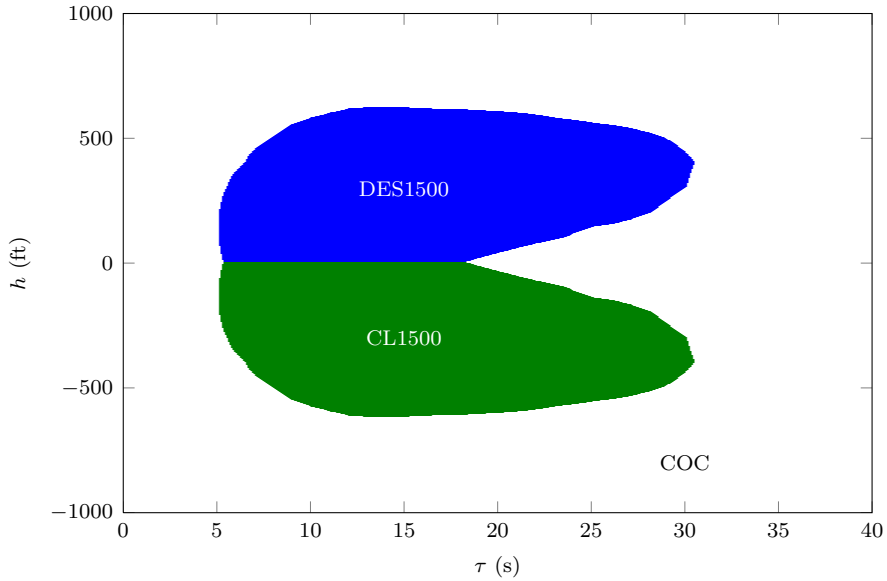
4.3 ROBUSTNESS TO MODEL PARAMETER VARIATION

This section analyzes the robustness of the optimized logic on the white-noise encounter model with varying process noise parameter. In the following discussion, the modeled noise parameter refers to $\sigma_{\ddot{h}}$ used when optimizing the logic, and the environment noise parameter refers to $\sigma_{\ddot{h}}$ used in the white-noise encounter model during evaluation.

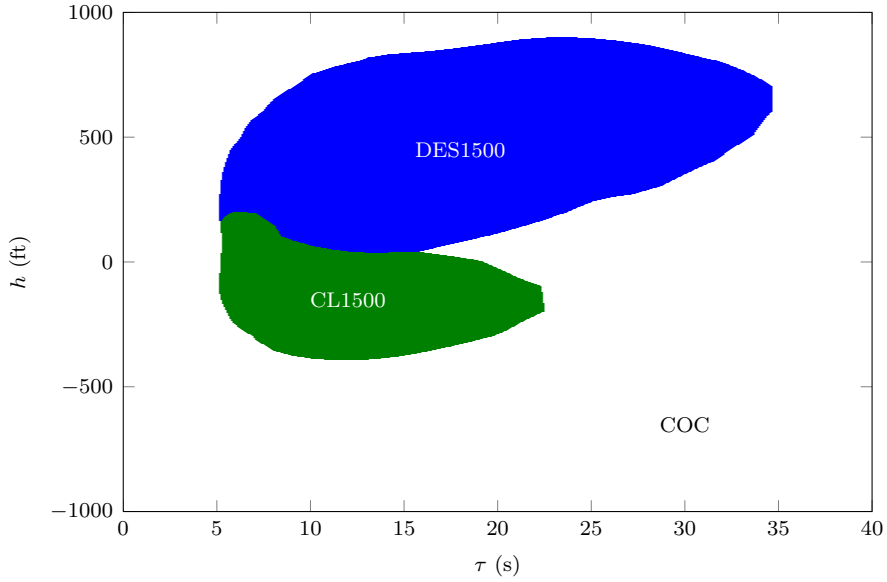
In the first set of experiments, the modeled noise parameter is kept fixed at $8\,\text{ft/s}^2$. Two policy slices for this setting of $\sigma_{\ddot{h}}$ are shown in Fig. 8. The environment noise parameter is varied from $0\,\text{ft/s}^2$ to $15\,\text{ft/s}^2$. The performance of both the optimized logic and TCAS are shown in Fig. 9. Each point on the curves was estimated from one million simulations.

Due to the way encounters are initialized, trajectories with zero acceleration noise will always result in an NMAC unless an advisory is issued. When the environment noise parameter is zero, both TCAS and the DP logic alert in every encounter and successfully prevent NMAC. As the environment noise parameter increases, the system predictability decreases and some encounter trajectories diverge, initially causing the alert rate to decrease and the NMAC rate to increase for both the TCAS logic and the DP logic. The DP logic reaches its lowest alert rate when the environment noise matches its optimized value of $8\,\text{ft/s}^2$.

Increasing the environment noise above the modeled noise degrades performance, but the logic remains robust regardless of whether the noise is over- or under-estimated. Regardless of the actual environment noise, the DP logic is sufficiently robust to significantly outperform TCAS.

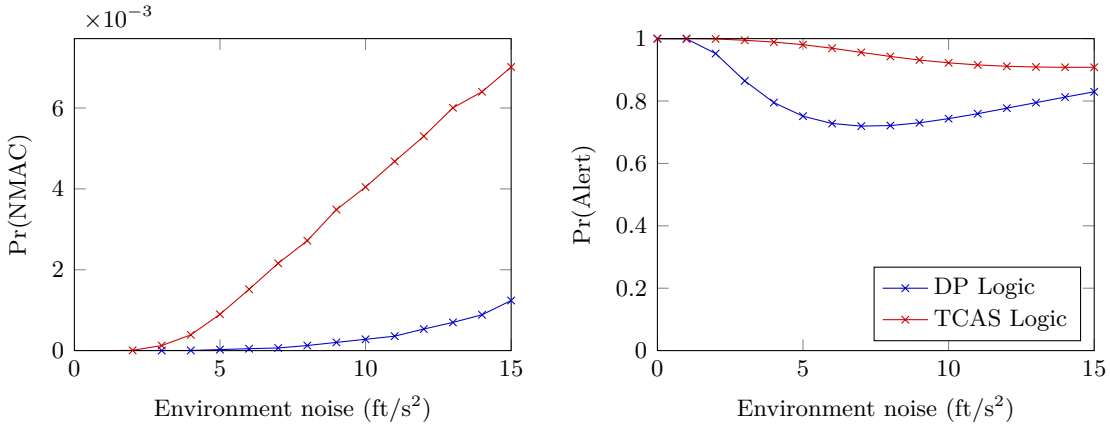


(a) $\dot{h}_0 = 0\,\text{ft/min}, \dot{h}_1 = 0\,\text{ft/min}, s_{\text{RA}} = \text{COC}$



(b) $\dot{h}_0 = 1000\,\text{ft/min}, \dot{h}_1 = 0\,\text{ft/min}, s_{\text{RA}} = \text{COC}$

Figure 8. Optimal action plots when $\sigma_{\dot{h}} = 8\,\text{ft/s}^2$.



(a) Safety performance.

(b) Alert performance.

Figure 9. Parametric robustness with varying environment and constant logic optimized at $\sigma_{\ddot{h}} = 8\,\mathrm{ft/s^2}$.

In the second set of experiments, the modeled noise was varied from $0\,\mathrm{ft/s^2}$ to $15\,\mathrm{ft/s^2}$, but the environment noise was kept fixed at $8\,\mathrm{ft/s^2}$. The performance is summarized in Fig. 10. Increasing the noise modeled by the logic resulted in more alerts and fewer NMACs. As in Fig. 9, the number of NMACs is nearly minimized when the modeled noise matches the environment. Divergence between the environment dynamics and the modeled dynamics slightly degrades performance. The performance of TCAS is constant because it is invariant to the modeled noise parameter.

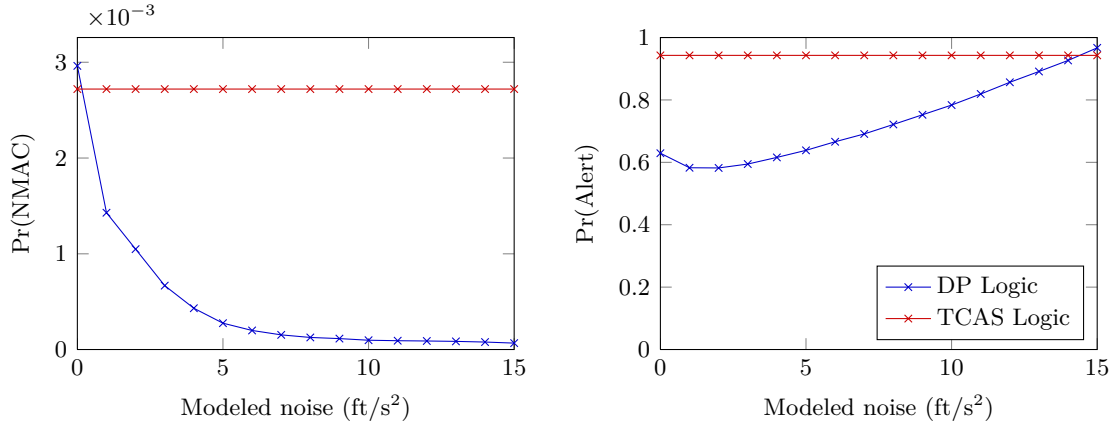
4.4 ROBUSTNESS TO MODEL STRUCTURE VARIATION

Differences between the environment and the model used by the logic may be more significant than simply variations in the process noise parameter. To determine the robustness of the system to structural inaccuracies in the modeled dynamics, both TCAS and the DP logic were evaluated on one million encounters generated by the correlated encounter model. Figure 11 shows the DP logic performance for two different costs of alerting.

The logic derived with an alert cost of 0.01 resulted in fewer NMACs than TCAS but a comparable number of alerts. When the alert cost was increased to 0.1, the DP logic was still more successful than TCAS in preventing NMACs while alerting much less frequently. The two DP logic curves show how choosing the alert cost can trade off safety for alert rate. These results indicate that even if the environment model is very different from the model used by DP to optimize the logic, the DP logic can still perform better than TCAS.

4.5 ROBUST DYNAMIC PROGRAMMING

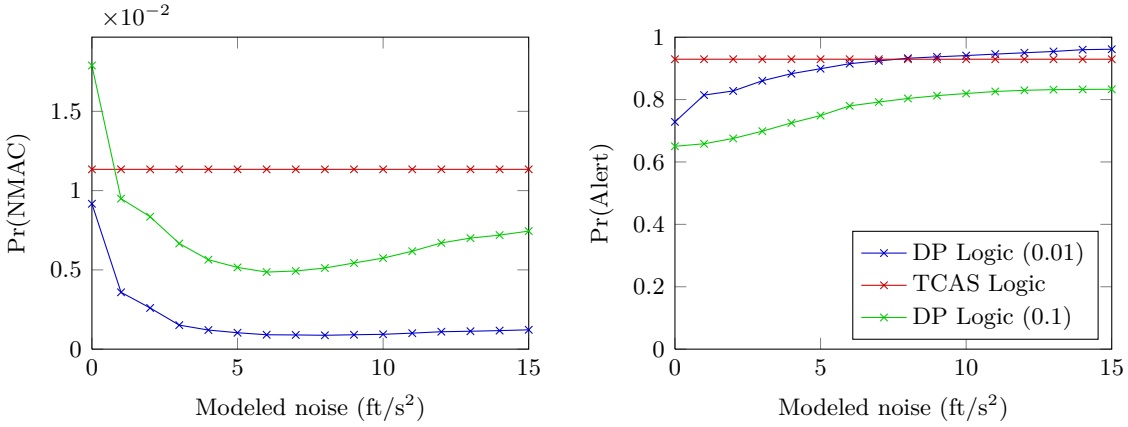
Robust dynamic programming (RDP) has been proposed as a technique for making the policies obtained through dynamic programming less sensitive to the particular choice of transition model



(a) Safety performance.

(b) Alert performance.

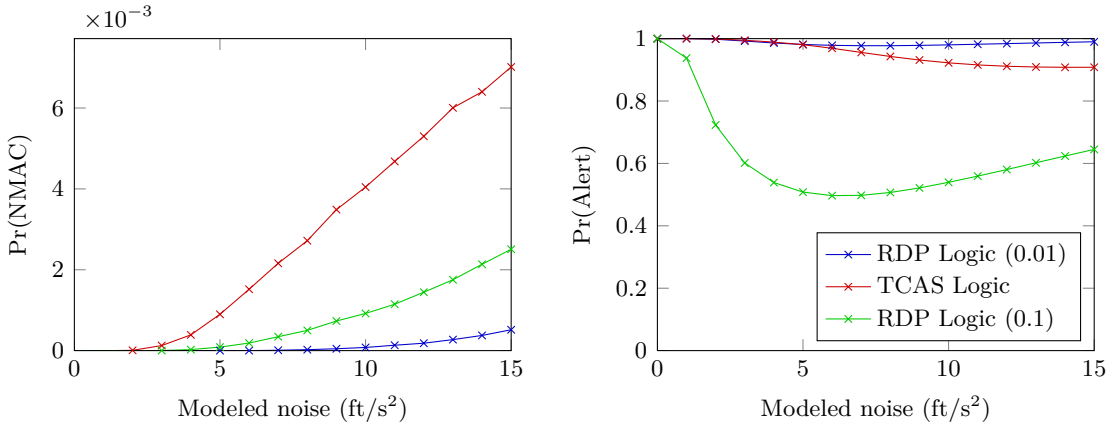
Figure 10. Parametric robustness with varying logic and constant environment using $\sigma_{\dot{h}} = 8\,\mathrm{ft/s^2}$.



(a) Safety performance.

(b) Alert performance.

Figure 11. Model structure robustness. DP alert cost is in parentheses.



(a) Safety performance.

(b) Alert performance.

Figure 12. Robust logic performance on white-noise encounter model. DP alert cost is in parentheses.

[42]. If the set of transition models is given by $T_1, \ldots, T_N$, then the value iteration algorithm is modified so that

$$J_k^*(s) = \min_a \max_i \left[C(s,a) + \sum_{s'} T_i(s,a,s') J_{k-1}^*(s') \right]. \tag{17}$$

Similarly,

$$J_k^*(s,a) = \max_i \left[C(s,a) + \sum_{s'} T_i(s,a,s') J_{k-1}^*(s') \right]. \tag{18}$$

To determine the effect of robust dynamic programming on the performance of the system, the logic was optimized using models with process noise parameters of $4\,\mathrm{ft/s^2}$, $6\,\mathrm{ft/s^2}$, and $8\,\mathrm{ft/s^2}$. Figure 12 shows the results when the environment noise parameter of the white-noise encounter model was varied from $0\,\mathrm{ft/s^2}$ to $15\,\mathrm{ft/s^2}$. The resulting RDP logic significantly reduced the number of NMACs compared to TCAS but did so at the expense of more alerts. Logic derived with an alert cost of 0.1 cut the alert probability in half while maintaining the NMAC rate well below that of TCAS. Compared to the DP logic optimized for a single model (Fig. 9), the RDP logic provides a flatter response, reducing the sensitivity of the optimized logic to environment noise.

The robust logic was also evaluated on the correlated encounter model. Table 5 summarizes the results. The DP logic optimized using an alert cost of 0.01 achieves a minimum NMAC rate of 0.000875, as shown in Fig. 11, lower than that of TCAS (0.0113), while alerting at approximately the same rate. The RDP logic using the same alert cost of 0.01 is able to reduce the NMAC rate even further (0.000295) while only marginally increasing the alert rate. By increasing the alert cost, the RDP logic has the potential to reduce the alert rate by 10 % and the NMAC rate by an order of magnitude compared to TCAS.

TABLE 5

Robust logic performance on correlated encounter model

Logic	Alert Cost	Pr(NMAC)	Pr(Alert)
TCAS	N/A	$1.13 \cdot 10^{-2}$	$9.29 \cdot 10^{-1}$
DP	0.01	$8.75 \cdot 10^{-4}$	$9.32 \cdot 10^{-1}$
RDP	0.01	$2.95 \cdot 10^{-4}$	$9.63 \cdot 10^{-1}$
RDP	0.10	$2.90 \cdot 10^{-3}$	$8.31 \cdot 10^{-1}$

4.6 STATE-BASED ROBUSTNESS ANALYSIS

The results in this section have been using Monte Carlo to estimate the NMAC and alert rates assuming a distribution over initial states. Each run of one million encounters took approximately 10 min on a 3 GHz processor, which is manageable when evaluating overall performance. However, one might be interested in estimating performance for a large collection of individual initial states. This is useful, for example, when identifying regions of the state space where the logic has difficulty resolving NMAC. Monte Carlo simulation may not, in such situations, be feasible. This section presents an alternative approach based on dynamic programming for efficiently evaluating metrics on the full state space. This approach can be used to identify regions of the state space where safety is degraded due to modeling error.

The objective is to compute $M(s)$, the expected value of some metric, for every starting state s in the discrete state space. The metric evaluated on a trajectory given by the state-action sequence $s_1, a_1, \ldots, s_N, a_N$ is calculated by

$$\sum_{n=1}^{N} m(s_n, a_n), \tag{19}$$

where m is the immediate metric function. For example, to estimate the probability of NMAC, $m(s, a)$ would be one if s is an NMAC state and zero otherwise.

Let π be the policy to be evaluated and T be the transition model used for evaluation. When evaluating the robustness of π, the transition model T need not be the model used for deriving π. An iterative process, similar to value iteration, is used to compute M from every discrete state. First, M_0 is set to zero for all states. The values for M_k are determined from M_{k-1} as follows:

$$M_k(s) = m(s, \pi(s)) + \sum_{s'} T(s, \pi(s), s') M_{k-1}(s'). \tag{20}$$

This iteration, as with value iteration, is repeated to the desired horizon.

Computing the probability of NMAC for the full discrete state space requires eight seconds on a single processor. Running Monte Carlo on the 8.7 million states would require 17 years, even if each state only required one minute instead of ten.

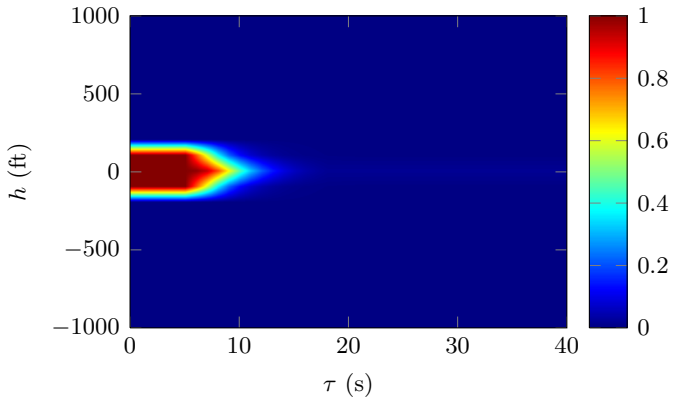
Figure 13 shows the probability of NMAC across the state space for selected environment noise values. The NMAC region expands and diffuses as the environment noise increases. The 5 s

pilot response delay causes the region for $\tau < 5\,\mathrm{s}$ to be uniform in the noiseless plot. The uniform and gradual transition of NMAC probability across the entire range of h and τ demonstrates the robustness of the logic across the state space.

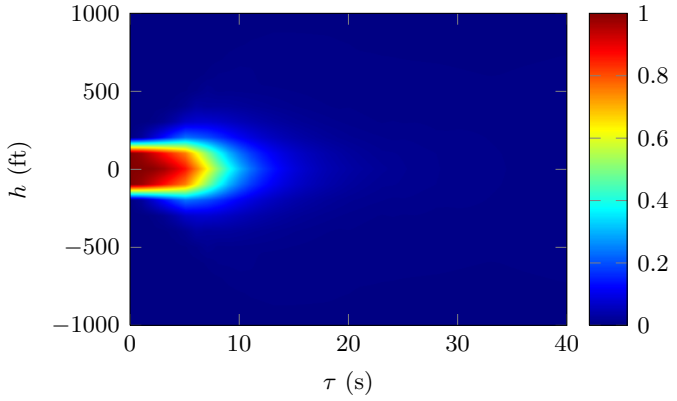
4.7 DISCUSSION

ATC-360 showed that dynamic programming results in logic that provides the lowest possible alert rate for a specified safety level under the assumption that the model used for optimization is correct. The results in this section demonstrate that even when the model used for optimization is inaccurate, the logic can still perform significantly better than TCAS in terms of preventing near mid-air collisions and reducing the alert rate. This section also explored the use of robust dynamic programming to further improve the robustness of the optimized logic to modeling errors, and experiments showed reduced sensitivity to both parametric and structural modeling errors. A state-based robustness analysis demonstrated logic robustness across the entire state space.

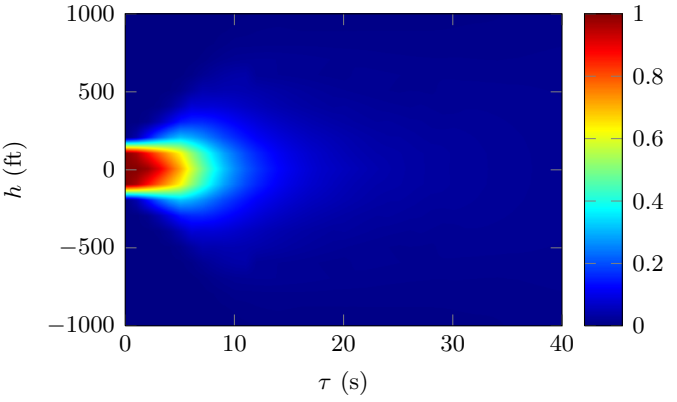
The robustness analysis in this section used encounter models. Although the encounter models were useful in generating large collections of encounters to stress test the system, further analysis will involve evaluating the logic robustness on recorded radar data. One complication with using recorded radar data is that there are relatively few near mid-air collisions in the airspace, in part due to the effects of TCAS. However, evaluation of the optimized logic on recorded encounters can aid in tuning the model and choosing an appropriate cost function. Even if the model is not perfectly tuned due to the sparseness of the recorded radar dataset, the optimized logic should still perform well as shown by the experiments in this section.



(a) $0\,\mathrm{ft/s^2}$ environment noise.



(b) $8\,\mathrm{ft/s^2}$ environment noise.



(c) $15\,\mathrm{ft/s^2}$ environment noise.

Figure 13. Probability of NMAC across the discrete state space evaluated at fixed environment noise values.

This page intentionally left blank.

5. COLLISION AVOIDANCE IN THREE SPATIAL DIMENSIONS

The previous section applied the logic optimized for a simple two-dimensional model (Section 3) to three-dimensional encounters. Although the optimized logic performed better than TCAS, it was limited by its inability to adequately model the horizontal dynamics. One way to better model encounters in three spatial dimensions is to add additional state variables. However, naively discretizing this (greatly expanded) state space would significantly increase the computational and storage requirements.

This section presents a general approach for approximating solutions to problems where only some of the state variables are controllable. In the collision avoidance problem, only the state variables representing the vertical motion of the aircraft are controllable. It is assumed that the state variables governing the horizontal motion of the aircraft evolve independently of the decisions made by the collision avoidance system. The approximation method involves decomposing the problem into two separate subproblems, one controlled and one uncontrolled, that can be solved independently offline using dynamic programming. During execution, the results from the offline computation are combined to determine the approximately optimal action from the current state. After explaining the general approximation method, which may be applied to a variety of different problems, this section applies it to collision avoidance.

5.1 PARTIAL CONTROL ASSUMPTIONS

It is assumed that the state is represented by a set of variables, some controlled and some uncontrolled. The state space of the controlled variables is denoted S_c, and the state space of the uncontrolled variables, S_u. The state of the controlled variables at time t is denoted $s_c(t)$, and the state of the uncontrolled variables at time t, $s_u(t)$. The solution technique may be applied when the following three assumptions hold:

1. The state $s_u(t+1)$ depends only upon $s_u(t)$. The probability of transitioning from s_u to s'_u is given by $T(s_u, s'_u)$.
2. The immediate cost $c(t+1)$ depends only upon $s_c(t)$ and $a(t)$. If the controlled state is s_c and action a is executed, the immediate cost is denoted $C(s_c, a)$.
3. The episode terminates when $s_u \in G \subset S_u$ with immediate cost $C(s_c)$.

Figure 14 shows the influence diagram for this model.

In the collision avoidance problem, s_c represents the state of the vertical motion variables, and s_u represents the state of the horizontal motion variables. The first assumption is satisfied because the advisories issued by the collision avoidance system do not influence the horizontal motion. The second assumption is satisfied because the immediate nonterminal cost only depends on the advisory state and the advisory being issued.

The third assumption requires the episode to terminate when s_u enters G. In this problem, G is the set of states where there is a horizontal NMAC, defined to be when an intruder comes

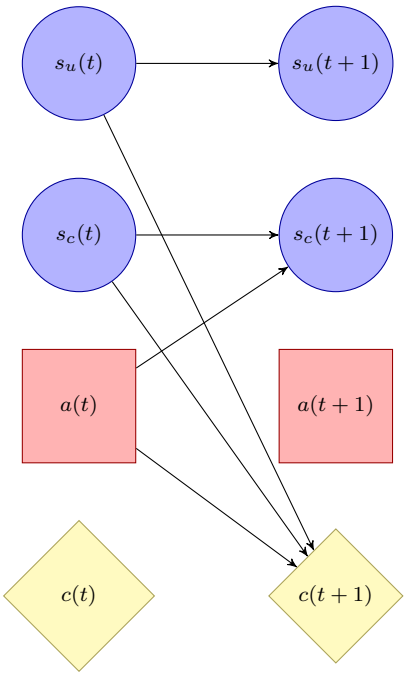


Figure 14. Influence diagram illustrating partial control in a Markov decision process.

within 500 ft horizontally. The immediate cost when this occurs is given by $C(s_c)$, which is one when the intruder is within 100 ft vertically and zero otherwise. In simulation, the episode does not necessarily terminate when s_u enters G, since entering G does not necessarily imply that there has been an NMAC (e.g., the two aircraft may have safely missed each other by 1000 ft vertically). However, it is generally sufficient to plan up to the moment where s_u enters G because adequate separation at that moment generally indicates that the encounter has been resolved.

5.2 CONTROLLED SUBPROBLEM

Solving the controlled subproblem involves computing the optimal policy for the controlled variables under the assumption that the time until s_u enters G, denoted τ, is known. In the collision avoidance problem, τ is the number of steps until another aircraft comes within 500 ft horizontally. Of course, τ cannot be determined exactly from $s_u(t)$ because it depends upon an event that occurs in the future, but this will be addressed by the uncontrolled subproblem (Section 5.3).

The expected cost from s_c given τ is denoted $J_\tau(s_c)$. The series $J_0, \ldots, J_K$ is computed recursively, starting with $J_0(s_c) = C(s_c)$ for all controlled states and iterating as follows:

$$J_k(s_c) = \min_a \left[C(s_c, a) + \sum_{s_c'} T(s_c, a, s_c') J_{k-1}(s_c') \right]. \tag{21}$$

The expected cost from s_c when executing a for one step and then following the optimal policy is given by

$$J_k(s_c, a) = C(s_c, a) + \sum_{s'_c} T(s_c, a, s'_c) J_{k-1}(s'_c). \tag{22}$$

The K-step expected cost when $\tau > K$ is denoted $J_{\bar{K}}$. It is computed by initializing $J_0(s_c) = 0$ for all states and iterating Eq. (21) to horizon K. The series $J_0, \ldots, J_K, J_{\bar{K}}$ is saved in a table in memory, requiring $O(K|A||S_c|)$ entries.

For the collision avoidance problem, the tables were computed offline in less than two minutes on a single 3 GHz Intel Xeon core using a horizon of $K = 39$ steps. Storing only the values for the valid state-action pairs requires 263 MB using a 64 bit floating point representation. For the experiments in this section, the same model and cost parameters assumed in Section 3 were used, except the cost of alerting was decreased to 0.001.

5.3 UNCONTROLLED SUBPROBLEM

Solving the uncontrolled subproblem involves using the probabilistic model of the uncontrolled dynamics to infer a distribution over τ for each uncontrolled state s_u. This distribution is referred to as the entry time distribution because it represents the distribution over the time for s_u to enter G. The probability that s_u enters G in τ steps is denoted $D_\tau(s_u)$ and may be computed using dynamic programming. The probability that $\tau = 0$ is given by

$$D_0(s_u) = \begin{cases} 1 & \text{if } s_u \in G, \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

The probability that $\tau = k$ for $k > 0$ is computed from D_{k-1} as follows:

$$D_k(s_u) = \begin{cases} 0 & \text{if } s_u \in G, \\ \sum_{s'_u} T(s_u, s'_u) D_{k-1}(s'_u) & \text{otherwise.} \end{cases} \tag{24}$$

Depending on s_u, there may be some probability that s_u does not enter G within K steps. This probability is denoted $D_{\bar{K}}(s_u)$ and may be computed from $D_0, \ldots, D_K$:

$$D_{\bar{K}}(s_u) = 1 - \sum_{k=0}^{K} D_k(s_u). \tag{25}$$

The sequence $D_0, \ldots, D_K, D_{\bar{K}}$ is stored in a table with $O(K|S_u|)$ entries. Multilinear interpolation of the distributions may be used to determine $D_\tau(\mathbf{x}_u)$ at an arbitrary continuous state $\mathbf{x}_u$.

The dynamics of the aircraft in the horizontal plane are assumed to follow those of the white-noise encounter model described in Section 4.1. The motion can be described by a three-dimensional model, instead of the typical four-dimensional (relative positions and velocities) model, due to rotational symmetry. The three state variables are as follows:

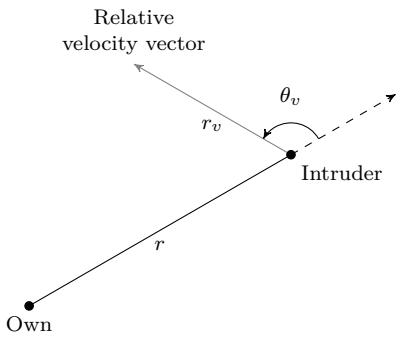


Figure 15. Three-variable model of horizontal dynamics.

TABLE 6

Uncontrolled variable discretization

Variable	Grid Edges
r	$0, 50, \ldots, 1000, 1500, \ldots, 40000\,\text{ft}$
r_v	$0, 10, \ldots, 1000\,\text{ft/s}$
θ_v	$-180°, -175°, \ldots, 180°$

- r: horizontal range to the intruder,
- r_v: relative horizontal speed, and
- θ_v: difference in the direction of the relative horizontal velocity and the bearing of the intruder.

These variables are illustrated in Fig. 15.

The entry time distribution can be estimated offline using dynamic programming. The state space was discretized using the scheme in Table 6, resulting in 730,000 discrete states. The offline computation required 92 s on a single 3 GHz Intel Xeon core. Storing $D_0, \ldots, D_{39}$ in memory using a 64 bit floating point representation requires 222 MB. Storing $D_{\bar{39}}$ is unnecessary because it can be inferred from the other tables. The standard deviation of the noise in the horizontal accelerations was set to $3\,\text{ft/s}^2$ when generating the tables for the experiments in this section.

An alternative to using DP for computing the entry time distribution offline is to use Monte Carlo to estimate the entry time distribution online. A Monte Carlo approach does not require the uncontrolled variables to be discretized and does not require $D_0, \ldots, D_K, D_{\bar{K}}$ to be stored in memory. However, using Monte Carlo increases the amount of computation required online. Since the NMAC region is small in the collision avoidance problem, the number of samples required to produce an adequate distribution may be large. Importance sampling and other sampling methods may be used to help improve the quality of the estimates of the entry time distribution [13]. The experiments in this section use 100 trajectory samples to estimate the Monte Carlo entry time distribution.

Another alternative to DP is to use the simple point estimate of τ, originally suggested in Section 4. The range rate $\dot{r}$ may be computed directly from the horizontal state variables:

$$\dot{r} = r_v \cos(\theta_v). \tag{26}$$

If the aircraft are converging in range, then τ can be approximated by $-r/\dot{r}$. Otherwise, τ is set beyond the horizon. This approach to estimating τ is very fast, but it does not take into account the uncertainty of the estimate. In the experiments, this method is referred to as the simple entry distribution.

Figure 16 shows two slices of the entry distribution computed using DP, Monte Carlo, and the simple method. The plots show the mean of the entry distribution for different x and y horizontal displacements between the aircraft conditioned on certain values for the relative horizontal speed, r_v. The relative horizontal velocity is pointing directly left. The plots can be understood by imagining the own aircraft as lying stationary at the origin and the axes as indicating the location of the intruder. The DP entry distribution is more diffuse and smoother than that produced by Monte Carlo. The simple entry distribution has a mean under the horizon for much of the plot. It tends to overestimate the collision risk. Use of the DP entry distribution over the simple entry distribution, which only uses range and range rate similar to TCAS, allows for automatic horizontal miss distance filtering.

5.4 ONLINE SOLUTION

After $J_0, \ldots, J_K, J_{\bar{K}}$ and $D_0, \ldots, D_K, D_{\bar{K}}$ have been computed offline, they are used together online to determine the approximately optimal action to execute from the current state. For any discrete state s in the original state space, the expected cost $J^*_K(s, a)$ may be computed as follows

$$J^*_K(s, a) = D_{\bar{K}}(s_u) J_{\bar{K}}(s_c, a) + \sum_{k=0}^{K} D_k(s_u) J_k(s_c, a), \tag{27}$$

where s_u is the discrete uncontrolled state and s_c is the discrete controlled state associated with s. Combining the controlled and uncontrolled solutions online in this way requires time linear in the size of the horizon. Multilinear interpolation can be used to estimate $J^*_K(\mathbf{x}, a)$ for an arbitrary state $\mathbf{x}$, and from this the optimal action may be obtained.

The memory requirements for directly storing the true $J^*_K(s, a)$ is $O(|A||S_c||S_u|)$. However, the hybrid offline-online method presented in this section allows the solution to be represented using $O(K|A||S_c| + K|S_u|)$ storage, which can be a tremendous savings when $|S_c|$ and $|S_u|$ are large. For the collision avoidance problem, this method allows the cost table to be stored in 500 MB instead of over 1 TB. The offline computational savings are even more significant.

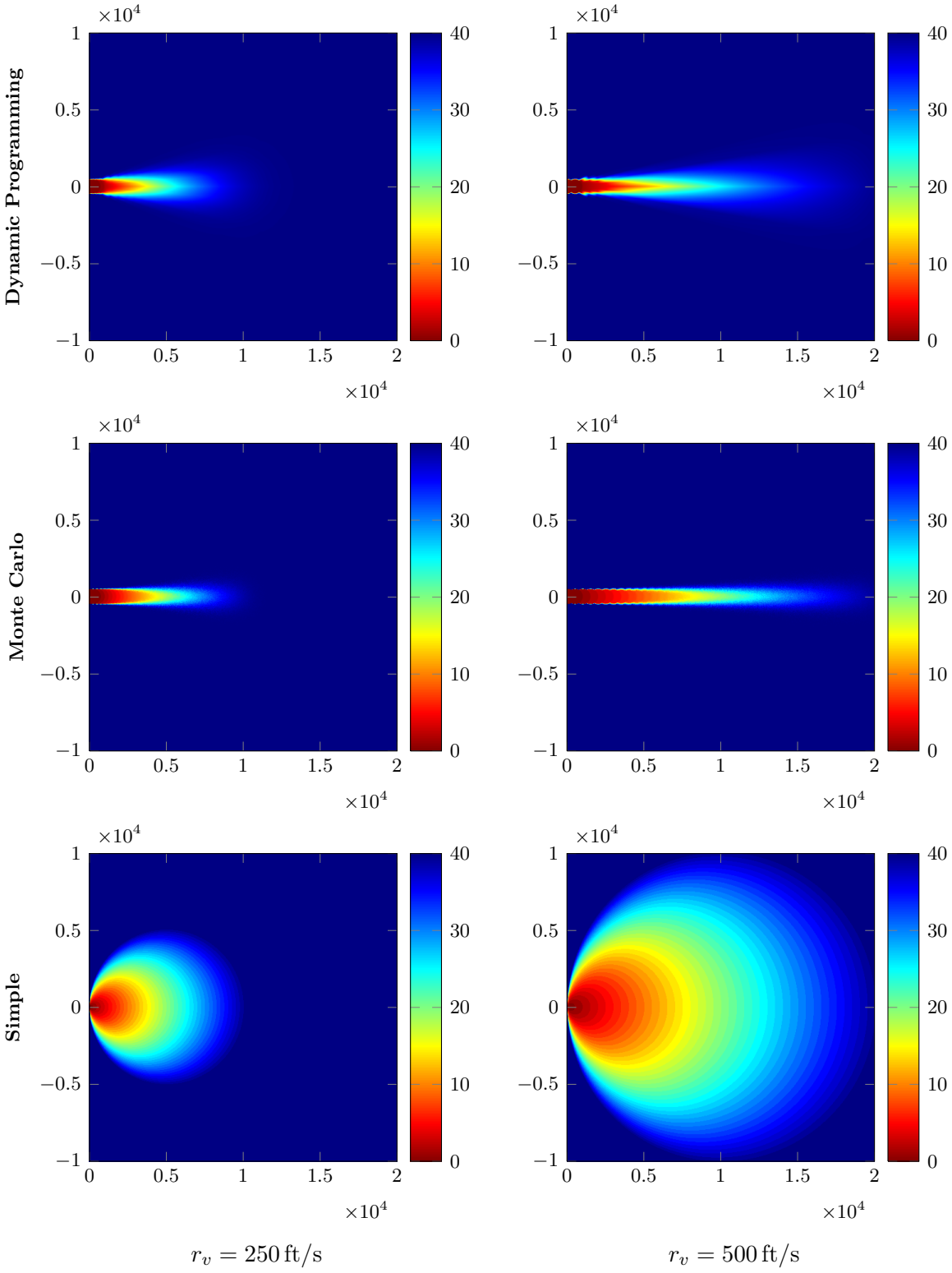


Figure 16. Mean of the entry distribution for two slices of the state space when the relative horizontal velocity is pointing directly left. Horizontal axis represents the relative x displacement (ft) and the vertical axis the relative y displacement (ft).

5.5 EXAMPLE ENCOUNTER

Figure 17 shows an example encounter comparing the behavior of the system using the DP entry time distribution against the TCAS logic. The encounter was produced using the white-noise encounter model. Figure 18 shows the entry time distribution computed using the three methods of Section 5.3 at the points in time when the system issues alerts.

Seventeen seconds into the encounter, the DP logic issues a descend to pass below the intruder. The expected cost for issuing a descend advisory is approximately 0.00928, lower than the expected cost for issuing a climb advisory (0.0113) or for not issuing an advisory (0.00972). The DP entry time distribution at this time has a conditional mean $\mathrm{E}[\tau \mid \tau < 40\,\mathrm{s}]$ of approximately 12.01 s. This represents the mean time until horizontal NMAC inside the horizon. A considerable portion of the probability mass ($\sim$40 %) is assigned to outside the horizon, when $\tau \geq 40\,\mathrm{s}$. The Monte Carlo entry time distribution, in comparison, has less support but a comparable conditional mean of 17.12 s. Only 15 % of the probability mass is concentrated on $\tau \geq 40\,\mathrm{s}$. The point estimate of τ using the simple method is 21.65 s.

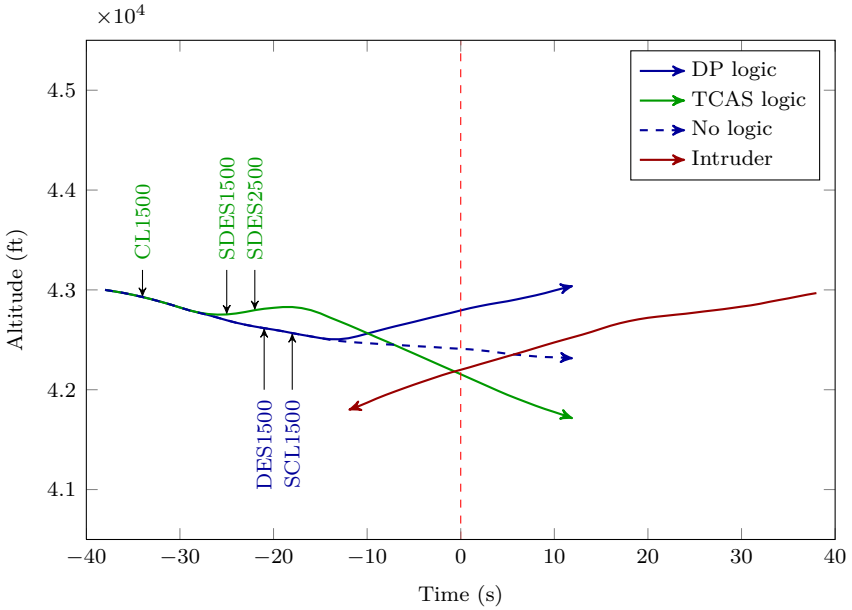
After the descend advisory is issued, the intruder begins to increase its descent, causing the DP logic to reverse the descend to a climb 20 s into the encounter. The pilot begins the climb maneuver three seconds later. Once the aircraft are safely separated, the DP logic discontinues the advisory at $t = 31\,\mathrm{s}$. The minimum horizontal separation is 342 ft, at which time the vertical separation is 595 ft. No NMAC occurs.

TCAS initially issues a climb advisory four seconds into the encounter because it anticipates, using straight-line projection, that by climbing it can safely pass above the intruder. Nine seconds later, when the own aircraft is executing its climb advisory, TCAS reverses the climb to a descend because it projects that maintaining the climb will not provide the required separation. TCAS strengthens the advisory three seconds later, but fails to prevent the NMAC. The aircraft miss each other by 342 ft horizontally and 44 ft vertically. Although the TCAS logic alerts earlier and more often, the DP logic still outperforms it in this example encounter.

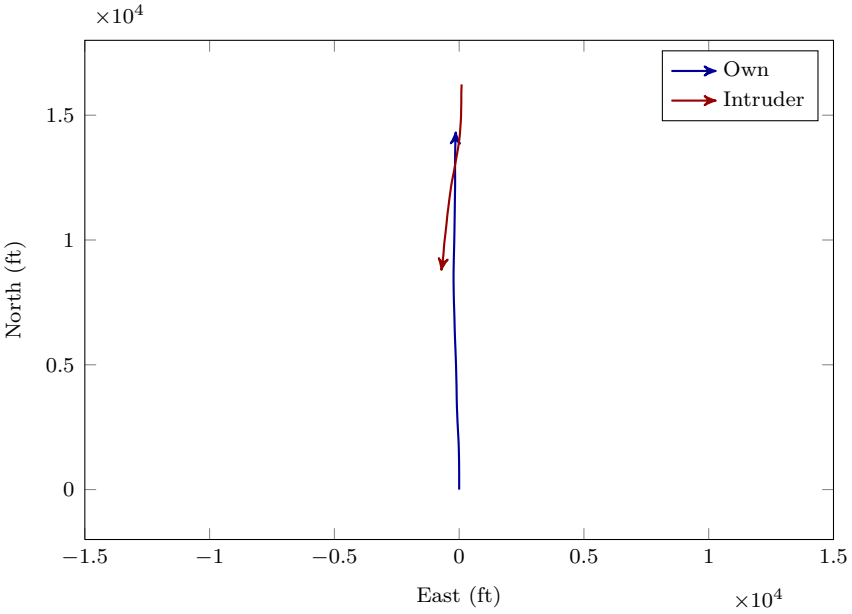
5.6 PERFORMANCE ASSESSMENT

Table 7 summarizes the results of simulating the DP logic and the TCAS logic on one million encounters generated by the white-noise encounter model of Section 4.1. The performance of the DP logic with the three entry distributions was assessed. The table summarizes the number of NMACs, alerts, strengthenings, and reversals.

As the table shows, the DP logic can provide a much lower NMAC rate while significantly reducing the alert rate. The Monte Carlo entry time distribution results in more NMACs, but it alerts less frequently than the other methods. Increasing the number of samples used generally improves performance but increases online computation time. The DP logic using the simple point estimate of τ resolves all but one NMAC while rarely reversing or strengthening the advisory, but alerts more frequently than Monte Carlo and DP.

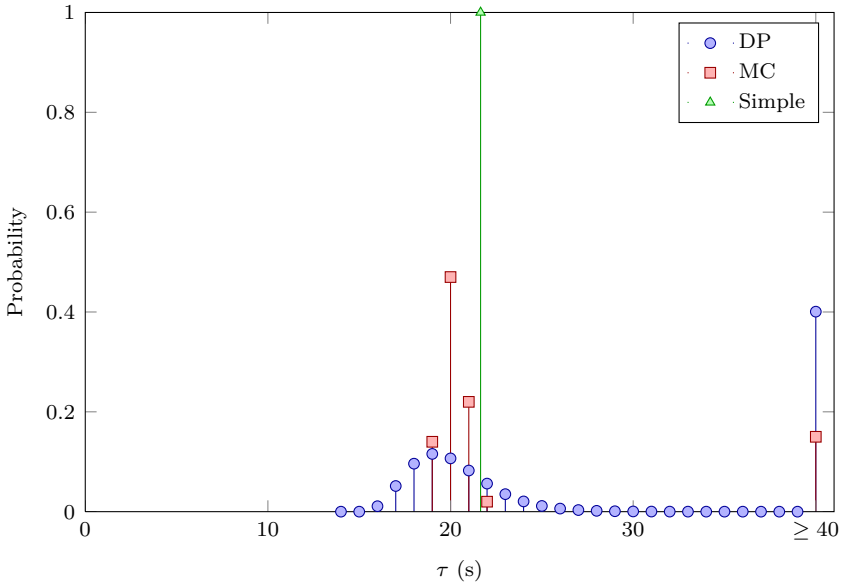


(a) Vertical profile.

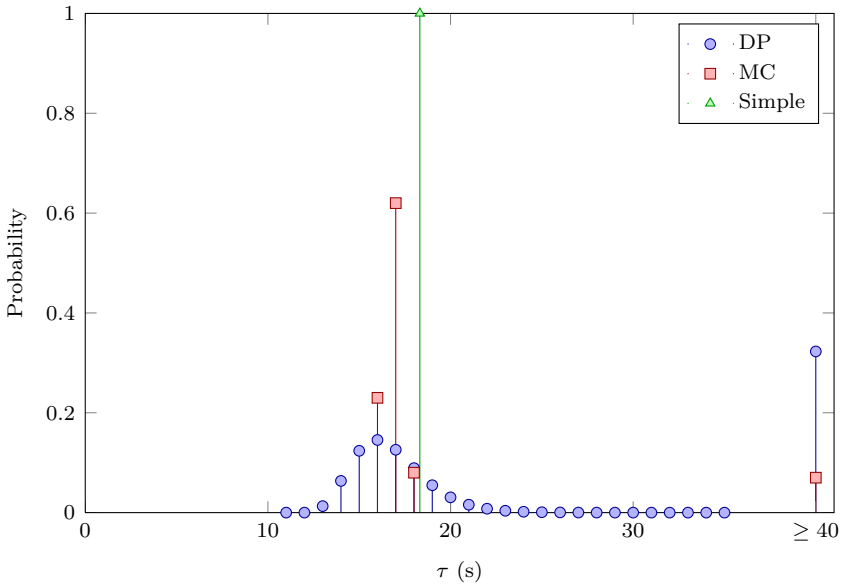


(b) Horizontal profile.

Figure 17. Example encounter comparing the system with the DP entry time distribution against TCAS.



(a) $t = 17\,\mathrm{s}$



(b) $t = 20\,\mathrm{s}$

Figure 18. Entry time distribution computed using dynamic programming (DP), Monte Carlo (MC), and the simple (Simple) methods at the two alerting points of the DP logic in the example encounter of Fig. 17.

TABLE 7

Three-dimensional performance evaluation

	DP Logic			
Metric	DP Entry	MC Entry	Simple Entry	TCAS Logic
NMACs	2	11	1	101
Alerts	540,113	400,457	939,745	994,640
Strengthenings	39,549	37,975	26,485	45,969
Reversals	1242	747	129	193,582

5.7 SAFETY CURVE

Figure 19 shows the safety curves for the DP logic and TCAS when different parameters are varied. Each point on the curves was estimated using 10,000 simulations from the white-noise encounter model. The DP logic safety curves were produced by varying the cost of alerting from zero to one while keeping the other event costs fixed. Separate curves were produced for the three methods of estimating the entry time distribution. The upper-right region of the plot corresponds to costs of alerting near zero and the lower-left region corresponds to costs near one.

The safety curve for TCAS was generated by varying the sensitivity level of TCAS. The sensitivity level of TCAS is a system parameter of the logic that increases with altitude. At higher sensitivity levels, TCAS will generally alert earlier and more aggressively to prevent NMAC.

The safety curves show that the DP logic can exceed or meet the level of safety provided by TCAS while alerting far less frequently. The safety curves can aid in choosing an appropriate value for the cost of alerting that satisfies a required safety threshold.

Figure 19 also reveals that the DP and Monte Carlo methods for estimating τ offer similar performance and that they both outperform the simple method, especially when the cost of alerting is high and the logic can only alert sparingly to prevent NMAC. In the upper-right region of the plot, the three methods are nearly indistinguishable. In this region, the systems always alert and are always successful in preventing NMAC.

5.8 TERMINAL CYCLING

Section 5.1 justified terminating encounters when the horizontal separation drops below 500 ft because usually at that point the encounter has resulted in NMAC or has been resolved. In roughly head-on encounters, the horizontal separation will typically drop below 500 ft for no more than one decision point. However, in slow-closure encounters where aircraft are flying nearly parallel to each other, the aircraft may come within 500 ft for multiple decision points. In such situations, it may be desirable to issue an advisory to prevent collision.

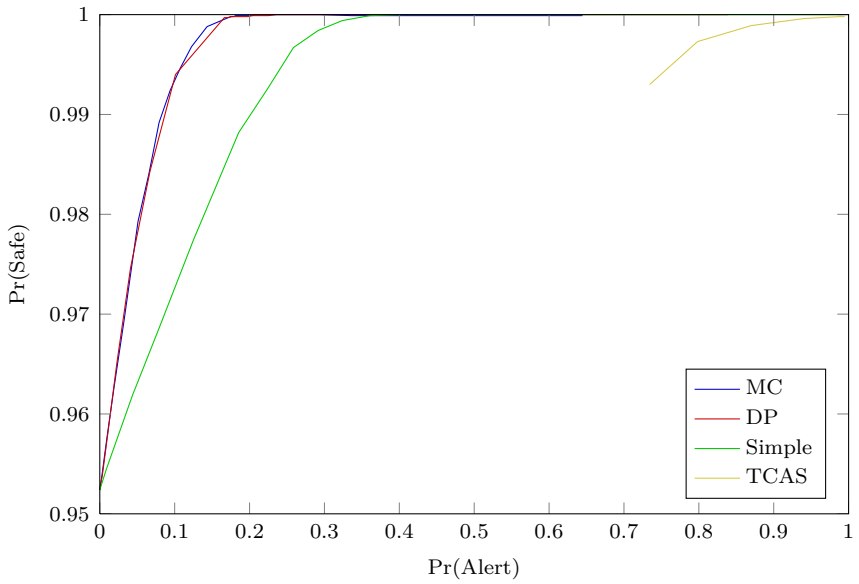


Figure 19. Safety curves. Each point on the curves was estimated from 10,000 simulations.

One remedy for this situation is to incorporate cycles at states where $\tau = 0$. By cycling at $\tau = 0$ up to some horizon, it allows the system to model the effect of advisories after the intruder comes within 500 ft or some other chosen threshold. Consequently, the optimized logic will issue alerts even when $\tau = 0$ because it can result in preventing collision.

One way to efficiently compute $J_0(s_c)$ with N cycles is to first initialize $J_0(s_c) = C(s_c)$ and then apply the update rule

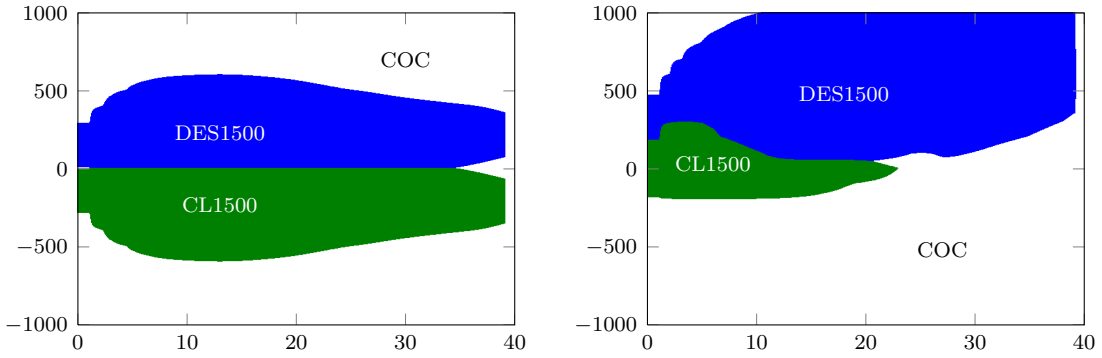
$$J_0(s_c) \leftarrow \min_a \left[C(s_c) + \sum_{s'_c} T(s_c, a, s'_c) J_0(s'_c) \right] \tag{28}$$

N times. Once J_0 is obtained, $J_1, \ldots, J_K, J_{\bar{K}}$ may be computed as before using the procedure in Section 5.2.

Figure 20 shows example policy plots using five terminal cycles. The alert region shifts entirely to the left. The remainder of this report uses five cycles, corresponding to the typical five-second initial pilot response delay.

5.9 DISCUSSION

The experiments demonstrate that the collision avoidance logic that results from solving the MDP using the method presented in this report reduces the risk of collision by a factor of 50 while issuing fewer alerts than TCAS in the simulated encounters. The system reverses less than 1 % of the time that TCAS reverses, and the system strengthens less frequently as well. This section assumed a



(a) $\dot{h}_0 = 0\,\text{ft/min}, \dot{h}_1 = 0\,\text{ft/min}, s_{\text{RA}} = \text{COC}$ (b) $\dot{h}_0 = 1500\,\text{ft/min}, \dot{h}_1 = 0\,\text{ft/min}, s_{\text{RA}} = \text{COC}$

Figure 20. Optimal action plots for five terminal cycles. Horizontal axis indicates τ (s) and the vertical axis indicates h (ft).

deterministic pilot response to advisories, but the next section will show how the MDP can be easily modified to accommodate probabilistic responses.

The remainder of the report will use the decomposition approach introduced in this section to approximate the solution to the full three-dimensional MDP. Unless otherwise stated, dynamic programming will be used to estimate the entry time distribution.

6. PROBABILISTIC PILOT RESPONSE MODELS

Previous sections have discussed formulating the collision avoidance problem as a Markov decision process and solving for the optimal collision avoidance strategy using dynamic programming. The logic was optimized to a pilot response model in which the pilot responds deterministically to all advisories. The current TCAS logic also uses a deterministic pilot response model to predict the future paths of the aircraft. However, it is unlikely that in actual flight the pilot will respond to advisories exactly according to the assumed deterministic model. As recorded radar data have shown, there is significant variability in the delay and strength of the response of pilots to advisories [43]. This section extends the methodology of the previous sections to include probabilistic pilot response models that capture the variability in pilot behavior in order to enhance robustness.

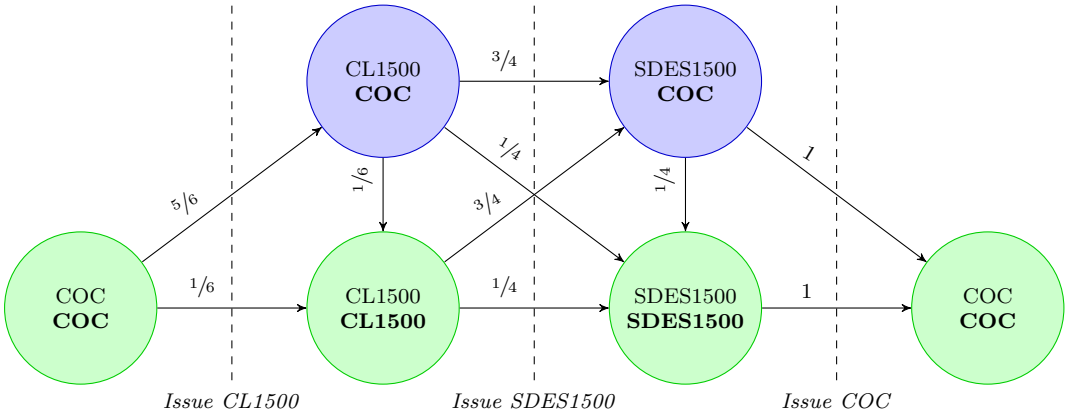
6.1 PILOT RESPONSE MODELS

Two models were constructed to capture the response of the pilot to resolution advisories. These models describe how the variable s_{RA} changes in response to different advisories. Each state in the Markov chain indicates the active advisory as well as the current response. The state "CL1500/COC," for example, signifies that the CL1500 advisory is currently on display but that the pilot is unresponsive to any advisory. In the first model, the number of states scales linearly with the number of advisories (13 states in total). The second model scales quadratically with the number of advisories (49 states in total). Figure 21, while not explicitly enumerating all state-action pairs, captures the essential features of the models.

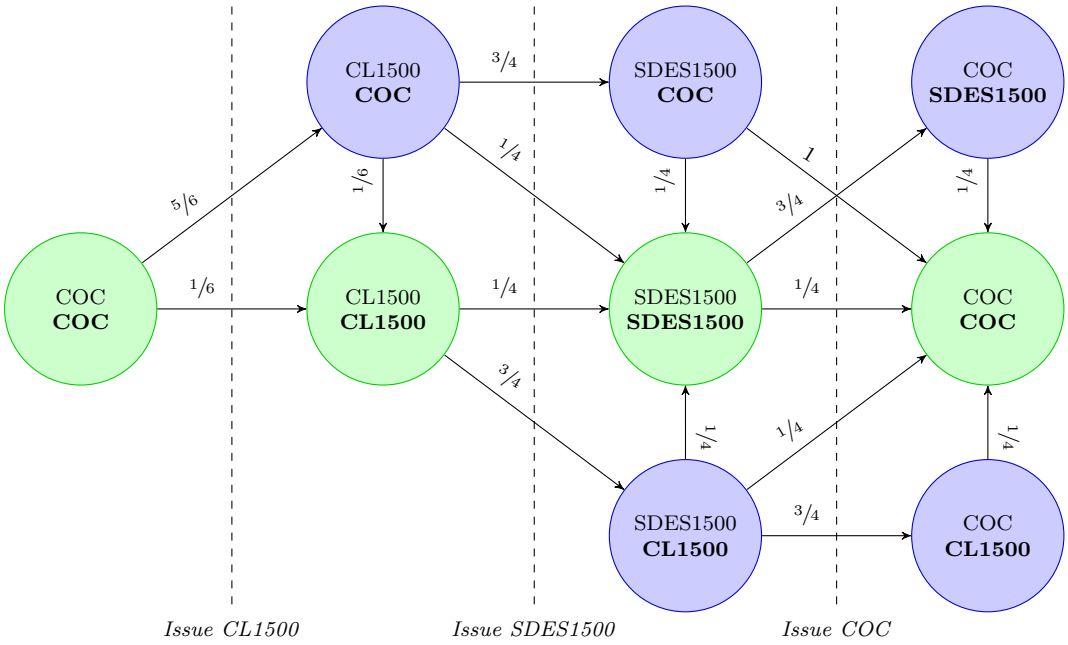
Before an advisory is issued, the Markov chain is in the "COC/COC" state. After a climb advisory is issued, for example, the pilot responds immediately with probability 1/6, transitioning to "CL1500/CL1500," and remains unresponsive otherwise, transitioning to "CL1500/COC." Should the climb advisory continue at the next time step, the pilot responds with probability 1/6 if he has not responded already. For a given advisory, therefore, the response delay is a geometric random variable; a success probability of 1/6 was chosen so that, on average, the pilot will respond in five seconds.

According to the linear model, if a descend advisory is subsequently issued, the pilot responds with probability 1/4 and neglects all advisories otherwise, regardless of whether he was responding to the climb advisory. The quadratic model differs in that, if the pilot is responding to the climb advisory, he will continue to respond to the climb advisory with probability 3/4. With a success probability of 1/4 the pilot will respond to the new advisory in three seconds on average.

If the advisory is discontinued, the Markov chain transitions to "COC/COC" with probability one in the linear model. However, in the quadratic model, the pilot retains some memory of the advisory he was previously executing and continues executing it with probability 3/4 even after the advisory is terminated.



(a) Linear model.



(b) Quadratic model.

Figure 21. Probabilistic pilot response models. The pilot response is in bold, and the active advisory is in regular text. Transition probabilities are indicated along the directed arcs. Cycles are omitted.

6.2 OPTIMAL POLICY

The optimal policy was computed using a unit NMAC cost, an alert cost of 0.01, a reversal cost of 0.01, a strengthening cost of 0.009, and a clear-of-conflict reward of 1×10^{-6}. Five terminal cycles were used. Figure 22 depicts two sections of the optimal policy when both aircraft are flying level, DES1500 is active, and the pilot is responding to it.

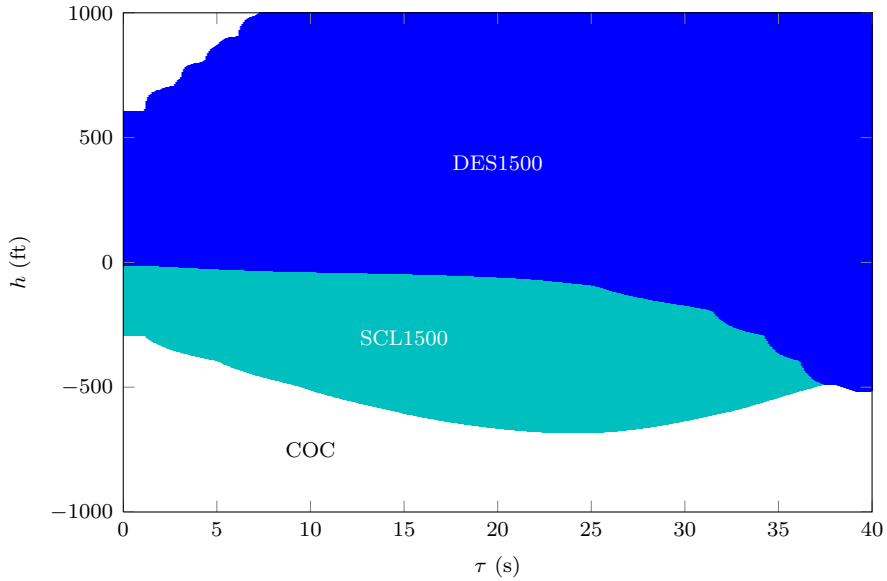
Figure 22(a) shows the optimal policy computed using the linear pilot response model. The blue region indicates the best action is to continue issuing the descend advisory. The descend advisory is typically maintained when the intruder is above the own aircraft. However, for large values of τ, the descend advisory is continued even when the own aircraft is above because there is sufficient time for the own aircraft to safely pass below the intruder. In the teal region, the optimal policy is to reverse the descend to a climb when the own aircraft is above the intruder, rendering the continued descent likely to induce NMAC. This region widens and narrows as τ changes. In certain regions of the state space, depicted in white, the best action is to discontinue the advisory.

Figure 22(b) is the optimal policy computed using the quadratic pilot response model. The policy is similar to that of the first plot. Because the pilot will continue to follow the descend advisory with probability 3/4 even after it is switched to a climb, the reversal region has been expanded to allow additional time for the pilot to reverse and prevent NMAC. The reversal region is smaller for the linear model because the pilot will become unresponsive to all advisories with probability 3/4 when the reversal is issued, which proves to be a much safer course of action than continuing the descent.

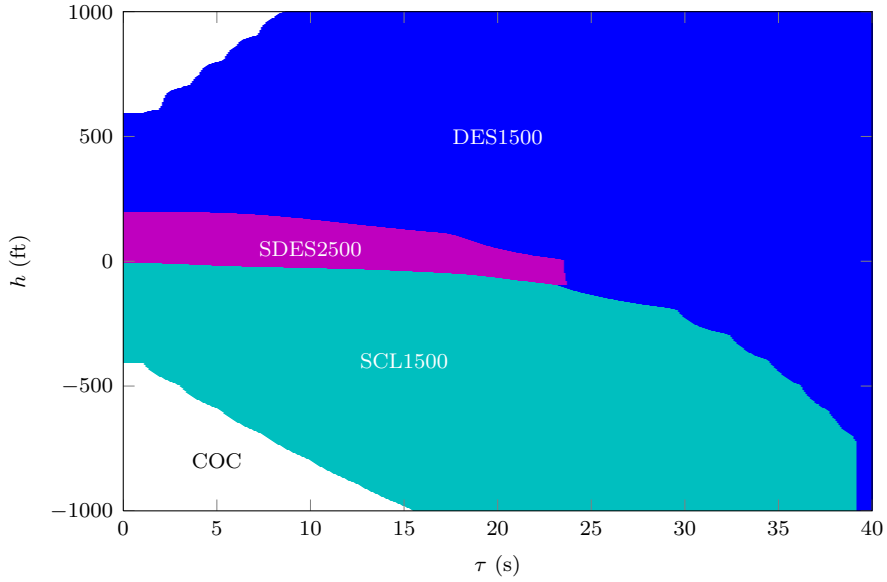
The purple region in Fig. 22(b) marks the places in the state space where the optimal action is to strengthen the descent. This region is absent in the first plot because the response of the pilot to strengthenings, together with the strengthening cost, makes continuing the advisory more advantageous than strengthening. Namely, because the pilot ignores all advisories with probability 3/4 and responds only with probability 1/4 after the strengthening is issued, it is unlikely that strengthening can improve safety, especially when NMAC is imminent. The probability of responding to the strengthening is also 1/4 in the quadratic model. However, because the pilot continues his descent with probability 3/4, the probability of strengthening, though the same, causes strengthening to have a lower expected cost than continuing the descend advisory.

6.3 BELIEF STATE FILTERING

During the online execution of the logic, determining the alert to issue at each time step requires knowledge of the state of the resolution advisory, s_{RA}. Unlike deterministic pilot response, even in the absence of sensor noise, there is uncertainty as to whether the pilot is responding to an advisory, thus making s_{RA} not fully observable. Instead, a probability distribution, or belief state, over possible values of s_{RA} must be maintained to summarize the beliefs regarding the response of the pilot to advisories. As new observations of the aircraft state are made each time step, the belief state $b(s_{\text{RA}})$ is recursively updated using standard model-based filtering techniques. The belief state is initialized at $t = 0$ to $b_0(\text{COC/COC}) = 1$. For all subsequent time steps $t > 0$, the



(a) Linear pilot response model.



(b) Quadratic pilot response model.

Figure 22. Optimal action plots for $\dot{h}_0 = 0\,\text{ft/min}$, $\dot{h}_1 = 0\,\text{ft/min}$, *and* s_{RA} = *DES1500/DES1500.*

belief state is updated as follows:

$$b_t(s'_{\mathrm{RA}}) \propto \sum_{s_{\mathrm{RA}}} p(\dot{h}'_0 \mid \dot{h}_0, s_{\mathrm{RA}}, a)\, T(s_{\mathrm{RA}}, a, s'_{\mathrm{RA}})\, b_{t-1}(s_{\mathrm{RA}}), \tag{29}$$

where $p(\cdot \mid \dot{h}_0, s_{\mathrm{RA}}, a)$ represents the probability density of the own aircraft vertical rate at time t conditioned on the previously observed vertical rate, $\dot{h}_0$, the action taken, a, and the advisory state, s_{RA}. This density is evaluated at the observed vertical rate at the current time, $\dot{h}'_0$. After each iteration of the filtering process, the action to take accounts for the likelihood of different responses using the QMDP approximation [30]:

$$\pi^*(b) = \arg\min_a \sum_s b(s) J^*(s, a). \tag{30}$$

This filtering approach relies on a model of how pilots respond to advisories. If the model is not reflective of reality, the filtering may become unstable as the model fails to explain unexpected observations. It is important, therefore, to design and vet a pilot response model that captures a wide spectrum of pilot behavior.

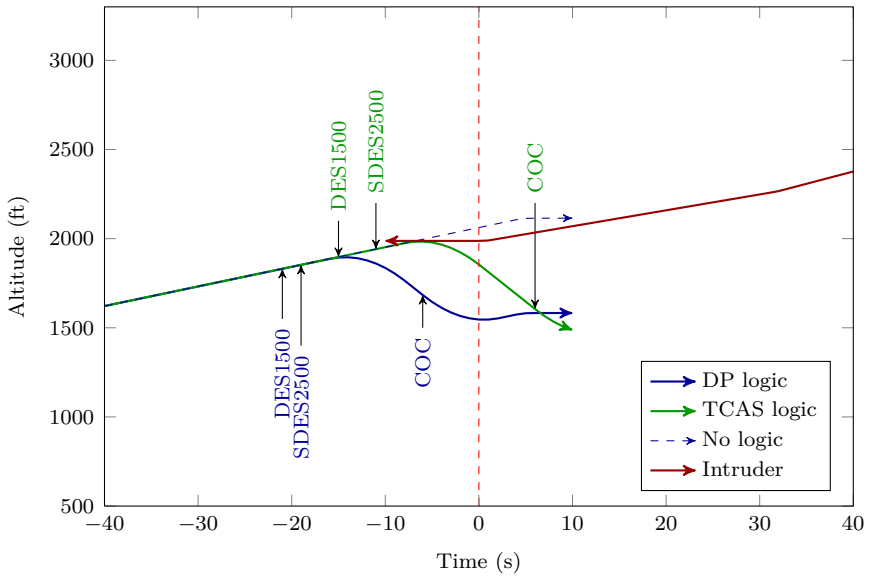
Advisory belief updating using a Bayesian framework is one way of monitoring the progression of the encounter. TCAS, similarly, makes accommodations to handle situations when the own aircraft is moving counter to the current advisory. In such a situation, if certain requirements are met, the advisory will be reversed to prevent potentially dangerous vertical chases. One strength of the current approach, however, is that the introduction of specialized heuristics is unnecessary.

6.4 EXAMPLE ENCOUNTER

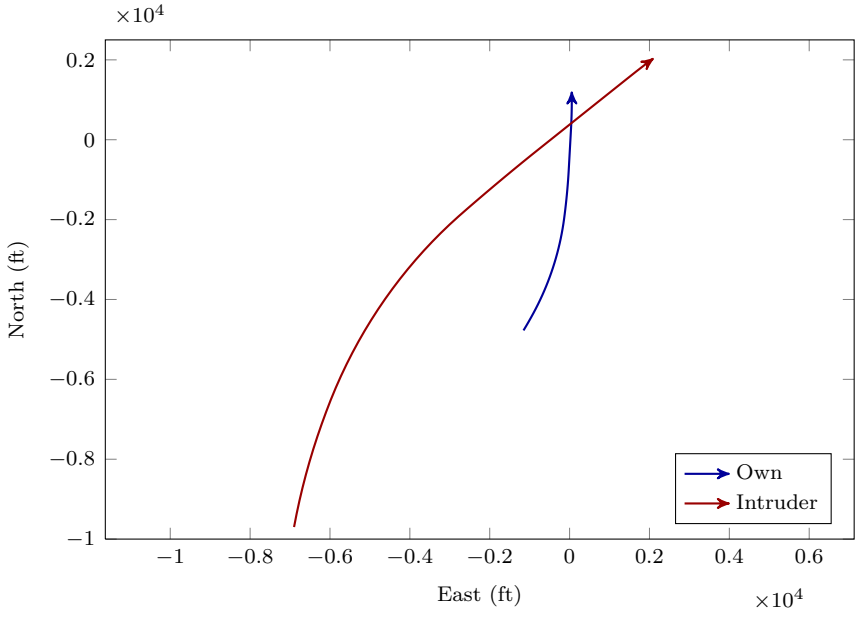
Figure 23 shows an example encounter comparing the behavior of the DP logic using probabilistic pilot response with that of TCAS. The logic was optimized using the quadratic pilot response model. The encounter was generated using the correlated encounter model. In this example encounter, although the logic is optimized using a probabilistic pilot response model, the pilot responds to all initial advisories in exactly five seconds and to all subsequent advisories in exactly three seconds.

Figure 24 shows the s_{RA} belief state throughout the course of the example encounter. Figure 25 shows the evolution of the entry time distribution. The entry time distribution was computed using the white-noise horizontal model with an acceleration standard deviation of $3\,\mathrm{ft/s^2}$ and an expanded horizontal NMAC region of 1000 ft.

Nineteen seconds into the encounter, the DP logic issues a descend to pass below the intruder. The expected cost for issuing a descend advisory is approximately 0.043, lower than the expected cost for issuing a climb advisory (0.054) or for not issuing an advisory (0.046). After the descend advisory is issued, as Fig. 24 illustrates, the probability distribution over s_{RA} indicates that with probability 5/6 the own aircraft is not executing the descend advisory and with probability 1/6 it is executing it. These are the probabilities of not executing and executing an initial advisory according to the model, respectively.



(a) Vertical profile.



(b) Horizontal profile.

Figure 23. Example encounter comparing the performance of the DP logic optimized using the quadratic pilot response model with that of TCAS.

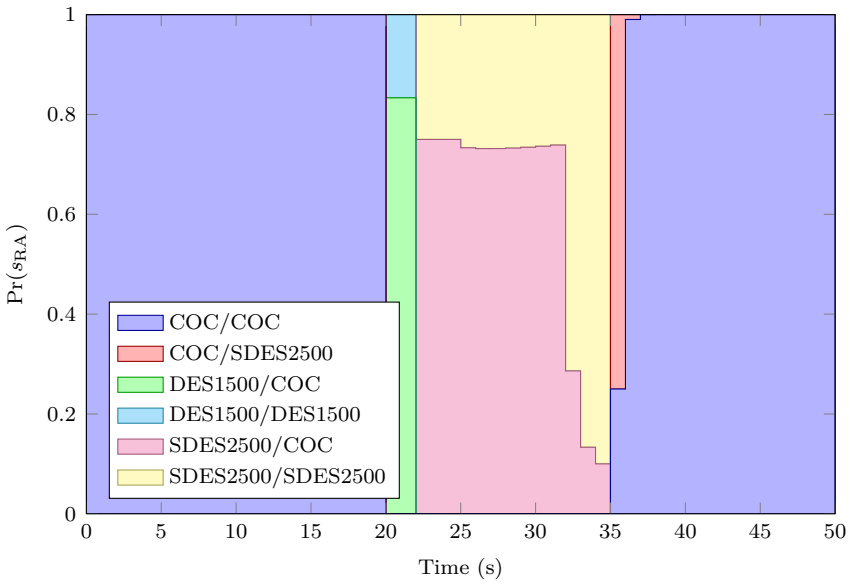


Figure 24. Advisory belief state over time for the example encounter.

Two seconds after the descend advisory is issued, the system strengthens the advisory. At the following time step, the probability distribution over s_{RA} reveals that the own aircraft is unresponsive to the strengthening with approximately 3/4 probability, which again represents the probability of not executing a subsequent advisory. With approximately 1/4 probability the own aircraft is responding to the strengthening and with a very small probability (on the order of 10^{-6}), not shown, the own aircraft is following the initial descend advisory. As the own aircraft begins to respond to the strengthening three seconds after its issuance, the distribution is updated to reflect this change in pilot behavior. Ten seconds after the pilot begins responding, it is believed that the own aircraft is responding to the strengthening with probability 0.9. The entry time distribution falls off quickly as the aircraft come within close proximity laterally.

The DP logic discontinues the advisory at $t = 34\,\mathrm{s}$, and the belief state over s_{RA} quickly changes in response to the own aircraft leveling off. The aircraft are vertically separated by 441 ft at the point of minimal horizontal separation (368 ft), and an NMAC does not occur.

TCAS also initially issues a descend advisory, six seconds after the DP logic, and then strengthens four seconds later. The alerts come too late, and an NMAC results at $t = 39\,\mathrm{s}$.

6.5 SIMULATION RESULTS

Table 8 summarizes the results of evaluating the DP and TCAS logics on 500,000 encounters generated by the correlated encounter model. The performance of the DP logic optimized using three different pilot response models—deterministic, linear, and quadratic—was assessed. These three systems, together with TCAS, were tested in three operating environments: one with a deterministic pilot response, one with a linear probabilistic pilot response, and one with a quadratic

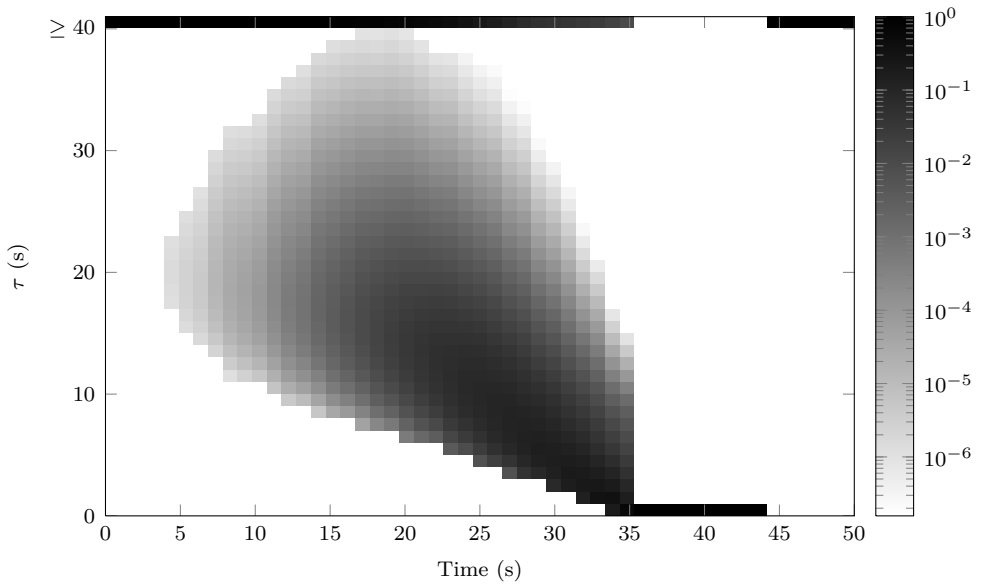


Figure 25. Entry time distribution over time for the example encounter.

probabilistic pilot response. The table compares the probabilities of certain events, such as an NMAC and an alert. The mean number of times the advisory is changed, E[RA changes], and the mean amount of time the systems spend alerting, E[RA duration], are also shown. Ensuring these two metrics stay low is critical to operational acceptability.

The following observations are worth mentioning:

1. The logic optimized through dynamic programming resolves more NMACs than TCAS while alerting less often. Although the DP logic does strengthen more frequently, the strengthening cost can be increased until a sufficiently low level of strengthening is achieved.

2. The logic computed using the deterministic model has an increased probability of NMAC when operating in a different environment than the one for which the logic is optimized. The logic computed using probabilistic pilot response appears less sensitive to unmodeled behavior.

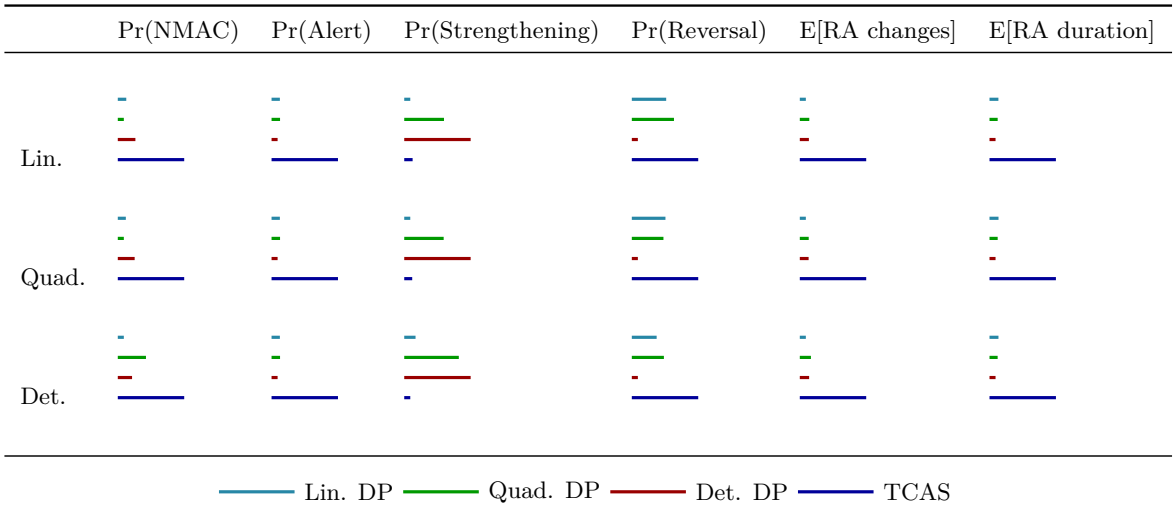
3. The linear and quadratic DP logics perform comparably in almost every category despite the structural differences in the pilot response models used to compute them. Quadratic DP strengthens more often, as might be expected based on Fig. 22.

The standard errors associated with the estimates of Table 8 were also calculated and were found to be small in relation to the actual estimates. The standard error, on average, was 10.08% the size of the estimate, indicating a reasonable level of accuracy in the estimation of the metrics.

Figure 26 shows the probability distributions for the number of advisory changes and the advisory duration for both the TCAS logic and the DP logic optimized using the linear pilot

TABLE 8

Performance metrics for different pilot response models

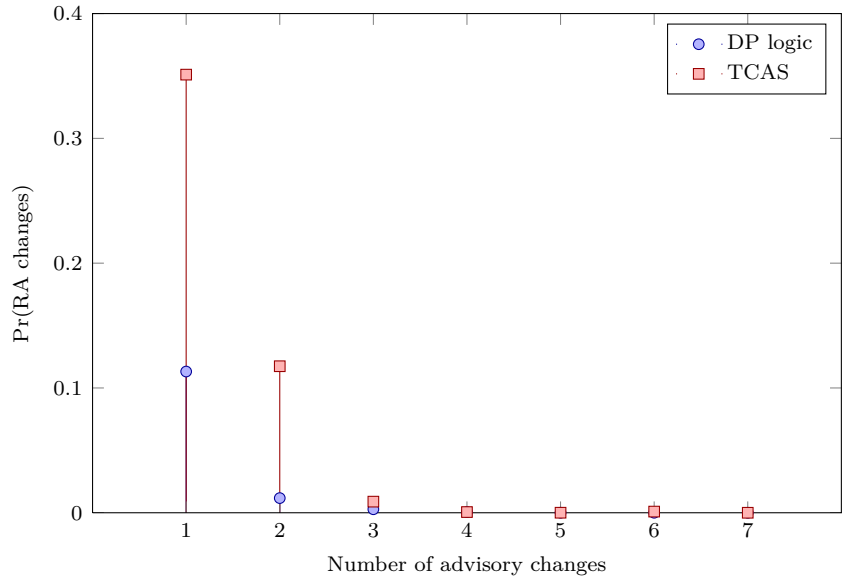


Note: The four bars in each cell of this table are normalized so that the relative performance of the four systems are more easily compared.

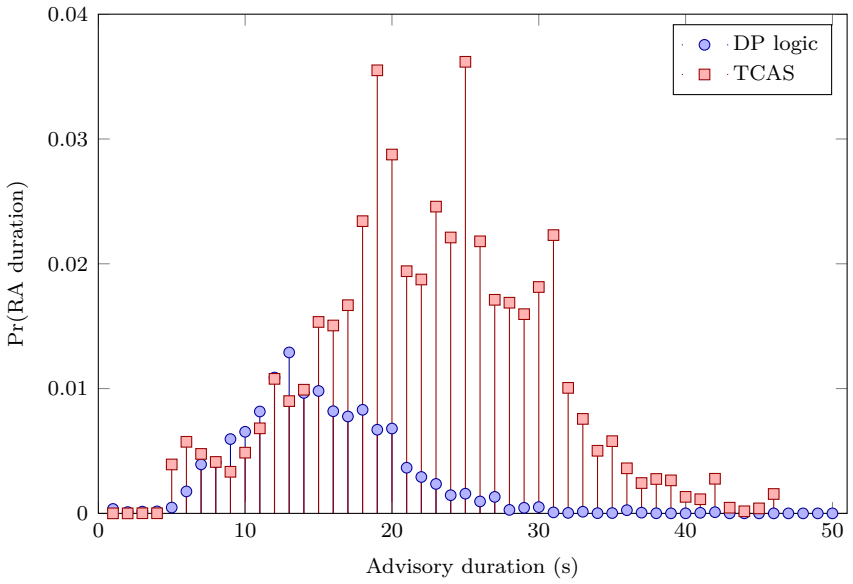
response model, evaluated on the linear pilot response model. Figure 26(a) shows that the DP logic changes the advisory less frequently than TCAS. Figure 26(b) indicates that the DP logic displays advisories for a shorter duration than TCAS.

6.6 DISCUSSION

This section presented extensions to the logic of the previous sections to include probabilistic pilot response models. Two Markov chain models were examined. Future work could explore the use of other models. In the two models presented in this section, when an advisory is issued, the pilot could respond to the advisory, continue following his current advisory (if any), or choose to ignore all advisories. They do not explicitly model situations in which the pilot runs counter to the advisory on display or situations in which the pilot may take a course of action vastly different than ones suggested by the advisories. They also do not model the pilot responding to advisories with a variety of different strengths. A collection of recorded radar data from TCAS-equipped aircraft can be used to construct a pilot response model that would be more representative of the actual response of pilots in the airspace.



(a) Distribution of advisory changes. The probability of no changes is 0.8645 for DP and 0.5211 for TCAS.



(b) Distribution of advisory duration. The probability of no duration is 0.8638 for DP and 0.5211 for TCAS.

Figure 26. Distributions of advisory changes and advisory duration for the DP logic optimized with linear pilot response and TCAS evaluated on the linear pilot response model.

7. STATE ESTIMATION

The previous sections discussed solutions to the collision avoidance problem in the absence of sensor error. Because real surveillance systems are imperfect, it is important to account for state uncertainty. As discussed in Section 2.2, solving for the optimal policy for a POMDP that accounts for imperfect observations generally requires approximation. This section applies the QMDP approximation method suggested in Section 2.3 to account for state uncertainty in a collision avoidance system.

As discussed earlier, the QMDP method uses the state-action costs $J_{\mathrm{MDP}}(s, a)$ over the underlying MDP states s to approximate the cost function $J(b, a)$ over belief states b. In particular,

$$J(b, a) = \sum_s b(s) J_{\mathrm{MDP}}(s, a). \tag{31}$$

That is, the expected cost when starting in belief state b, taking action a, and then continuing with the optimal policy is estimated by the expectation over the belief state of the state-action costs assuming no state uncertainty. Thus, under this approximation, the current uncertainty in the state, as encoded by b, is accounted for when choosing actions, but all future uncertainty is disregarded. Because the choice of advisory makes little difference in the reduction in state uncertainty, this approximation is expected to perform well.

7.1 SENSING

The sensor model used in this report is based on the current TCAS sensor. The sensor measures the slant range and bearing of all nearby intruder aircraft. The slant range error is modeled as a zero-mean Gaussian with 50 ft standard deviation. The bearing error is modeled as a zero-mean Gaussian with 10° standard deviation. There is no jitter in the intruder altitude measurements, but they are quantized at 25 ft intervals. It is assumed that the own altitude, vertical rate, and heading are provided by the onboard avionics without error. The model does not account for biases.

7.2 STATE ESTIMATION PROCESS

The belief state is updated recursively using all measurements up to and including the current time. Figure 27 outlines the steps in the state estimation process. Upon receipt of the first set of measurements, the state estimator, using the information provided in the first measurements and any prior knowledge about the state, computes the posterior state distribution, which reflects all present knowledge about the state given all information up to and including the present time. The state estimation component uses a sensor model together with a dynamic model to perform the estimation. As new information becomes available, a new posterior distribution is calculated, using the previous posterior distribution as the prior. State estimation using the Kalman filter follows this general flow. A Kalman filter performs Bayesian recursive estimation using Gaussian posterior distributions [44–46].

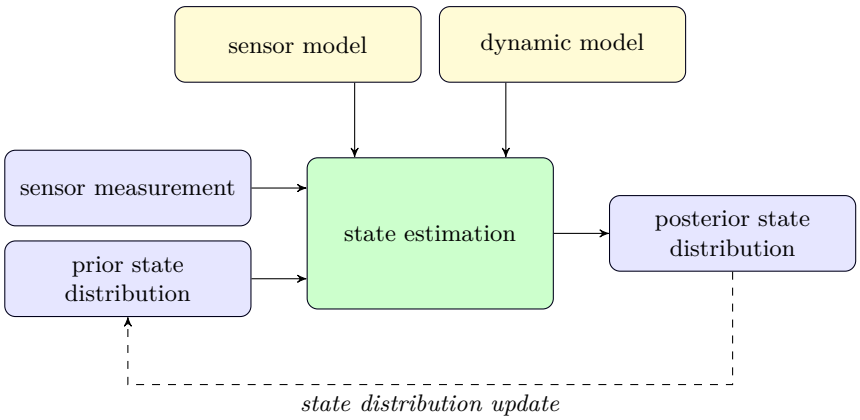


Figure 27. State estimation process.

The belief state is constructed using a composition of several different processes. Each process produces a collection of samples with associated probabilities representing different portions of the total belief state. The belief state is the Cartesian product of the collection of samples from all processes, representing the joint distribution over the following variables: h, $\dot{h}_0$, $\dot{h}_1$, s_{RA}, r, r_v, and θ_v. These individual processes are enumerated below.

1. Using observations of the slant range, bearing, and altitude of the intruder, as well as observations of the own altitude and heading, an unscented Kalman filter is used to estimate the Gaussian posterior distribution over the variables descriptive of the relative horizontal motion. Sigma-point samples from this distribution are drawn with associated weights, which are interpreted as probabilities. Each sample is converted into the three-dimensional representation consisting of r, r_v, and θ_v.

2. Another Kalman filter is used to estimate the Gaussian posterior of the intruder altitude and vertical rate given observations of intruder altitude. Because it is assumed that the own altitude and vertical rate are noiseless, no estimation is required for those quantities. The relative altitude h is estimated by the maximum *a posteriori* estimate of the intruder altitude from the Kalman filter minus the own altitude measurement. The intruder vertical rate $\dot{h}_1$ is estimated by the maximum *a posteriori* estimate of the intruder vertical rate from the Kalman filter. Because uncertainty in the intruder altitude and vertical rate is not projected to be significant, the maximum *a posteriori* estimates were used instead of the entire distribution in order to reduce computation time.

3. The own vertical rate is used to estimate the distribution over the advisory state variable s_{RA} according to Eq. (29). Because the posterior distribution of s_{RA} is discrete, the set of points on the support of the distribution serves as a collection of samples with their associated probabilities.

Further details regarding Kalman filtering as well as the details of the filters particular to this work can be found in Appendix B.

7.3 SIMULATION RESULTS

The DP logic, extended to account for state uncertainty using the QMDP method, was evaluated in simulation and compared against TCAS. Table 9 summarizes the results of simulating one million encounters using the white-noise encounter model. The white-noise model used $8\,\mathrm{ft/s^2}$ horizontal acceleration and $3\,\mathrm{ft/s^2}$ vertical acceleration standard deviations. The performance of DP using the DP and simple entry distributions is presented alongside TCAS. The performance of the systems without sensor noise serves as a baseline. During evaluation, the pilot responded deterministically to advisories.

The DP logic was computed using a unit NMAC cost, an alert cost of 0.001, a reversal cost of 0.01, a strengthening cost of 0.009, and a clear-of-conflict reward of 1×10^{-6}. Five terminal cycles were used. The logic was optimized using deterministic pilot response. The same entry distribution as in Section 6 was used.

The DP logic outperforms TCAS on all metrics except for strengthening. Even with the level of sensor noise expected by the current TCAS sensor, the DP logic still leads to greater safety while alerting less frequently. The DP logic also, on average, disrupts the pilot with fewer changes to the advisory and keeps advisories on display for a shorter period of time.

Table 10 summarizes the same performance metrics, this time estimated using the correlated encounter model. The DP logic was optimized using an alert cost of 0.01, keeping all other parameters the same. In realistic encounter scenarios with realistic levels of sensor noise, the DP logic is twice as safe as TCAS even though it alerts less than half the time. Figure 28 shows the convergence curves for several metrics from the tables.

7.4 SENSOR NOISE ROBUSTNESS

Because the sensor error may deviate significantly from the sensor error model assumed in Section 7.1, it is necessary for the logic to be robust to different degrees of state uncertainty. Figure 29 illustrates the effect bearing noise has on the probability of NMAC and of alert for both the DP logic and TCAS. The bearing sensor introduces the most noise in the state estimates, and it is used directly to estimate the DP entry distribution.

The TCAS logic is relatively insensitive to bearing noise because bearing measurements are only used to disable alerting in situations where the projected miss distance is large. The DP logic using the simple entry distribution is also relatively insensitive because it, like TCAS, uses range and range rate to estimate τ. The DP logic using the DP entry distribution is most sensitive to bearing noise because it uses bearing measurements to localize the intruder. In simulations using the correlated encounter model, performance is not significantly degraded with increased noise. The DP logic still results in fewer NMACs and fewer alerts than TCAS. The DP logic appears to be more sensitive to bearing error when evaluated on white-noise encounters. This is due to the fact that many of the white-noise encounters are head-on. A high level of sensor noise causes the system to misjudge the intruder heading and therefore not alert even when the intruder poses a significant threat.

TABLE 9

Performance evaluation on the white-noise encounter model

	TCAS Sensor			Perfect Sensor		
	DP/DP Entry	DP/Simple Entry	TCAS	DP/DP Entry	DP/Simple Entry	TCAS
Pr(NMAC)	$2.90 \cdot 10^{-5}$	$8.00 \cdot 10^{-6}$	$7.54 \cdot 10^{-5}$	$1.00 \cdot 10^{-6}$	$1.00 \cdot 10^{-6}$	$6.10 \cdot 10^{-5}$
Pr(Alert)	$7.51 \cdot 10^{-1}$	$9.00 \cdot 10^{-1}$	$9.94 \cdot 10^{-1}$	$7.16 \cdot 10^{-1}$	$9.91 \cdot 10^{-1}$	$9.94 \cdot 10^{-1}$
Pr(Strengthening)	$1.78 \cdot 10^{-1}$	$1.23 \cdot 10^{-1}$	$6.57 \cdot 10^{-2}$	$7.03 \cdot 10^{-2}$	$6.63 \cdot 10^{-2}$	$6.18 \cdot 10^{-2}$
Pr(Reversal)	$9.40 \cdot 10^{-3}$	$4.16 \cdot 10^{-3}$	$1.91 \cdot 10^{-1}$	$2.66 \cdot 10^{-3}$	$3.29 \cdot 10^{-4}$	$1.93 \cdot 10^{-1}$
E[RA changes]	$1.01 \cdot 10^{0}$	$1.10 \cdot 10^{0}$	$2.08 \cdot 10^{0}$	$8.31 \cdot 10^{-1}$	$1.02 \cdot 10^{0}$	$2.07 \cdot 10^{0}$
E[RA duration]	$1.15 \cdot 10^{1}$	$1.94 \cdot 10^{1}$	$3.96 \cdot 10^{1}$	$1.32 \cdot 10^{1}$	$2.65 \cdot 10^{1}$	$3.99 \cdot 10^{1}$

TABLE 10

Performance evaluation on the correlated encounter model

	TCAS Sensor			Perfect Sensor		
	DP/DP Entry	DP/Simple Entry	TCAS	DP/DP Entry	DP/Simple Entry	TCAS
Pr(NMAC)	$6.98 \cdot 10^{-5}$	$3.40 \cdot 10^{-5}$	$1.43 \cdot 10^{-4}$	$6.06 \cdot 10^{-5}$	$3.70 \cdot 10^{-5}$	$1.54 \cdot 10^{-4}$
Pr(Alert)	$2.01 \cdot 10^{-1}$	$3.92 \cdot 10^{-1}$	$5.03 \cdot 10^{-1}$	$1.16 \cdot 10^{-1}$	$4.02 \cdot 10^{-1}$	$4.66 \cdot 10^{-1}$
Pr(Strengthening)	$1.07 \cdot 10^{-1}$	$5.33 \cdot 10^{-2}$	$1.18 \cdot 10^{-2}$	$5.35 \cdot 10^{-2}$	$2.63 \cdot 10^{-2}$	$1.18 \cdot 10^{-2}$
Pr(Reversal)	$7.99 \cdot 10^{-4}$	$1.88 \cdot 10^{-3}$	$3.26 \cdot 10^{-3}$	$2.60 \cdot 10^{-4}$	$1.70 \cdot 10^{-3}$	$2.63 \cdot 10^{-3}$
E[RA changes]	$3.22 \cdot 10^{-1}$	$4.47 \cdot 10^{-1}$	$6.56 \cdot 10^{-1}$	$1.75 \cdot 10^{-1}$	$4.37 \cdot 10^{-1}$	$6.05 \cdot 10^{-1}$
E[RA duration]	$2.46 \cdot 10^{0}$	$8.14 \cdot 10^{0}$	$1.10 \cdot 10^{1}$	$1.48 \cdot 10^{0}$	$8.13 \cdot 10^{0}$	$1.04 \cdot 10^{1}$

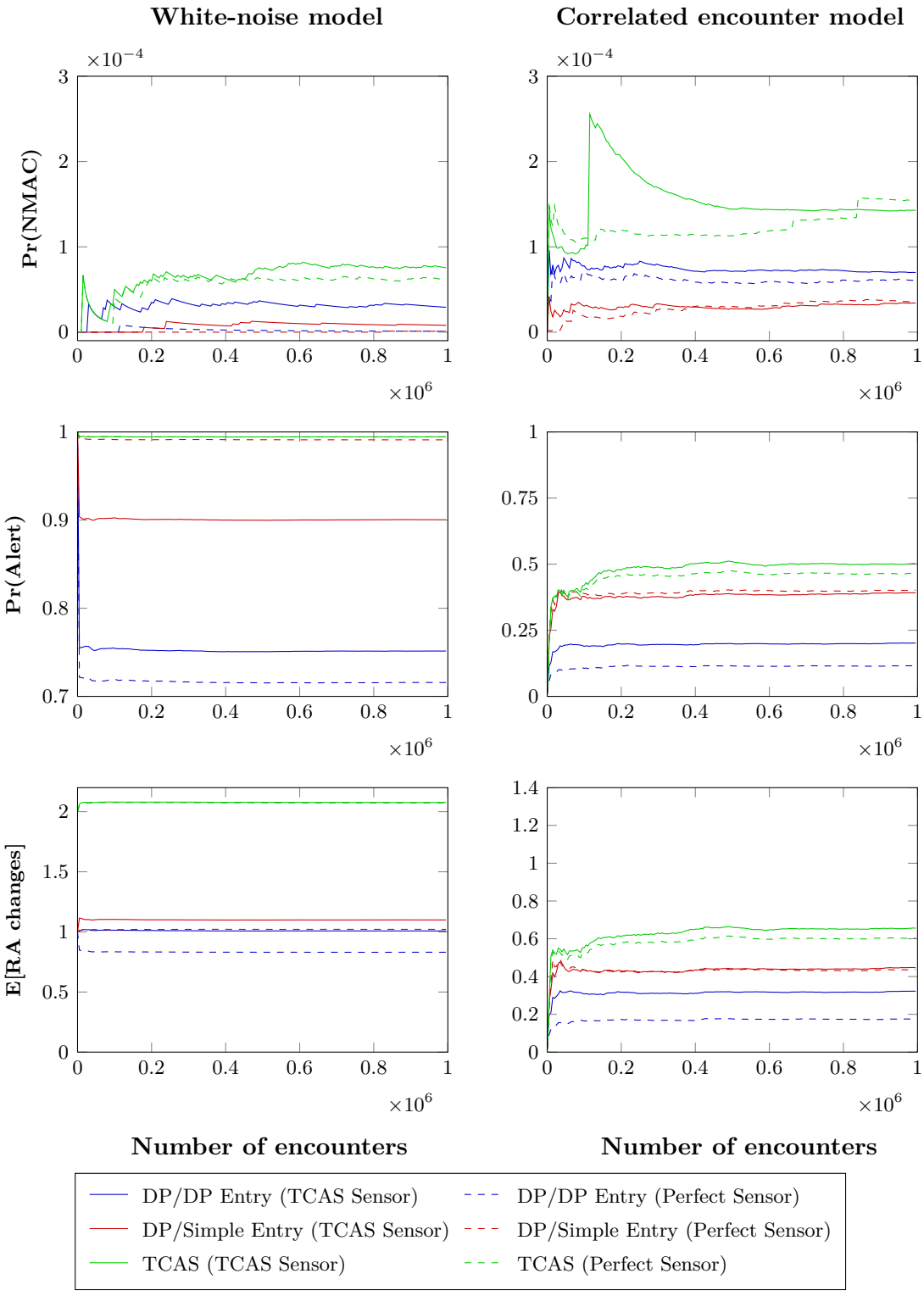


Figure 28. Convergence curves for DP and TCAS with and without sensor noise.

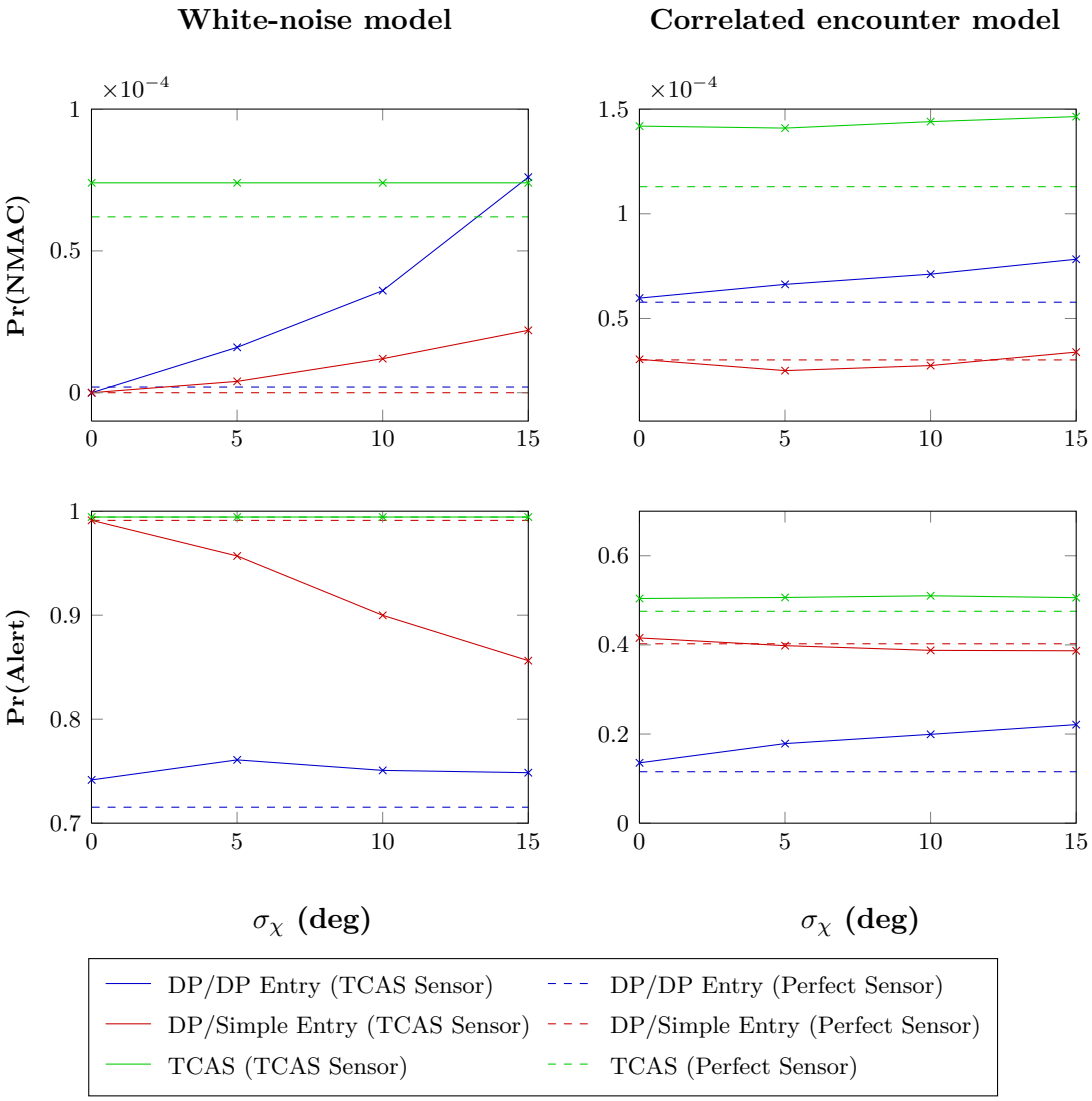


Figure 29. Sensor noise robustness. Each point on the curves was estimated from 500,000 simulations.

7.5 DISCUSSION

This section considered the extension of the logic presented in previous sections to account for state uncertainty using the QMDP method. The method was evaluated using the TCAS sensor with realistic sensor noise parameters. The results of simulations of realistic encounter scenarios suggest that the method works well, surpassing the performance of TCAS in terms of safety and other operational considerations.

Future work could explore the use of other approximation methods and sensor systems. For example, exploitation of GPS-based systems that have the potential of providing more accurate position estimates as well as velocities is likely to enhance the performance of the dynamic programming logic. The measurements from such systems can also be used to supplement the information currently provided by TCAS, thus improving its performance. It is important to note that the methodology proposed in this section can accommodate different sensor systems, as the solution of the MDP remains the same regardless of the particular sensor suite installed on the aircraft; no additional offline development is required.

This page intentionally left blank.

8. COORDINATION

The formulation of the collision avoidance problem in the previous sections assumed that the own aircraft, equipped with a collision avoidance system, encountered a single unequipped intruder. The collision avoidance system could influence the trajectory of the own aircraft by issuing advisories to the pilot, but it could not influence the intruder aircraft. In situations where both aircraft are equipped with collision avoidance systems, it is important that the advisories be coordinated to reduce the risk of inducing collision.

One way to model the problem of coordinating resolution maneuvers over a communication channel between multiple aircraft equipped with the same system is as a decentralized POMDP with communication (Dec-POMDP-Com), but solving them optimally is infeasible in general [47–49]. This section discusses a variety of different coordination strategies that differ in the policies they compute offline, their state estimation method, and how they select actions. These strategies are not necessarily optimal but they provide a lower NMAC rate and alert rate than the coordination strategy embedded in TCAS.

8.1 TCAS COORDINATION

TCAS coordinates with other TCAS-equipped aircraft. Upon selecting an advisory against a particular TCAS-equipped intruder, TCAS transmits a coordination interrogation to the intruder, called a Vertical Resolution Advisory Complement (VRC), through a dedicated communication channel restricting the set of advisories from which the intruder TCAS can choose. For example, if TCAS selects a climb advisory against the intruder, it sends a message to the intruder containing a "Do Not Climb" VRC. The VRC is used by the intruder to select a compatible advisory. When the intruder is no longer deemed a threat, TCAS sends a Cancel Vertical Resolution Advisory Complement (CVC) message canceling the previous VRC message. Also contained in the coordination message sent to the intruder is a parity field, called the Vertical Sense Bits (VSB) field. The intruder checks to see if the VSB field is consistent with the VRC and CVC fields before using the coordination information in the logic.

8.2 POLICY

Several different strategies may be used to formulate the MDP to be solved offline by DP. This section discusses three broad categories.

Joint Policy

A coordinated encounter between two aircraft with full state observability can be modeled as a multi-agent MDP (MMDP). An MMDP extends an MDP by allowing for a finite set of agents, each with a finite set of available actions. The joint action space is the Cartesian product of the action spaces of the individual agents. The state space is the set of all joint states, and the transition

and cost functions are defined over these joint state and action spaces. In an MMDP, each of the agents can fully identify the true state through the use of its own observations.

An MMDP can be reduced to a single-agent joint MDP over the joint state and action spaces. For a coordinated encounter, the action space is the set of all joint aircraft actions, such as DES1500 for one aircraft and CL1500 for the other. The state space is the set of all joint aircraft states. Extending the MDP introduced in Section 3 to an MMDP requires augmenting the advisory state variable s_{RA} to contain all information regarding the response of both pilots to their respective advisories. The cost function should also be amended so that cost is incurred when the other aircraft alerts, strengthens, or reverses. The MDP for the coordinated encounter generally consists of more states and actions than for the uncoordinated encounter. The solution to the joint MDP is a joint policy that specifies the actions for both aircraft from every state. So long as the size of the problem does not become too large, solving for the optimal joint policy can be done offline using DP.

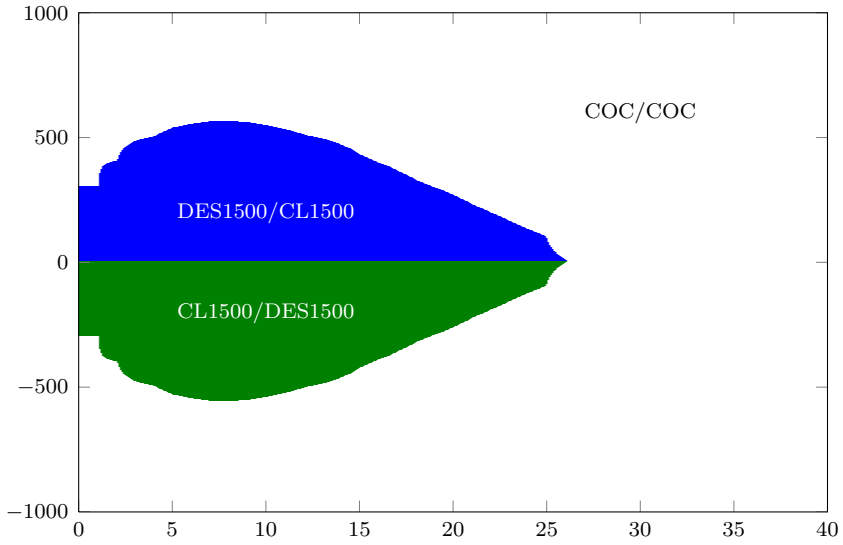
So long as computation and storage are not an issue, there is no harm in including in the joint action space all possible action pairs, including CL1500/CL1500. During the optimization, however, such incompatible action pairs would be assigned higher expected cost compared to other actions because they do not aid in reducing separation. Consequently, those incompatible actions would never be selected. To reduce the computational and storage requirements, these incompatible actions can be eliminated from consideration in the first place. Restricting the set of available joint advisories results in a smaller action space as well as a smaller state space. The number of states is reduced because the advisory state variable can assume fewer values.

The set of joint actions can be restricted even further by assuming that both aircraft issue initial alerts, reversals, and strengthenings at the same time. This reduced action set will be referred to as the synchronous action set and will be used to generate the results later in this section. The synchronous action set does not contain actions for which one of the aircraft alerts while the other one does not, although it might be beneficial to include such joint actions if it is possible to prevent NMAC reliably through the maneuvering of only one aircraft. A desirable property of the joint MDP with the synchronous action set is that it has the same number of states and actions as the MDP modeling the uncoordinated encounter when using the deterministic pilot response model.

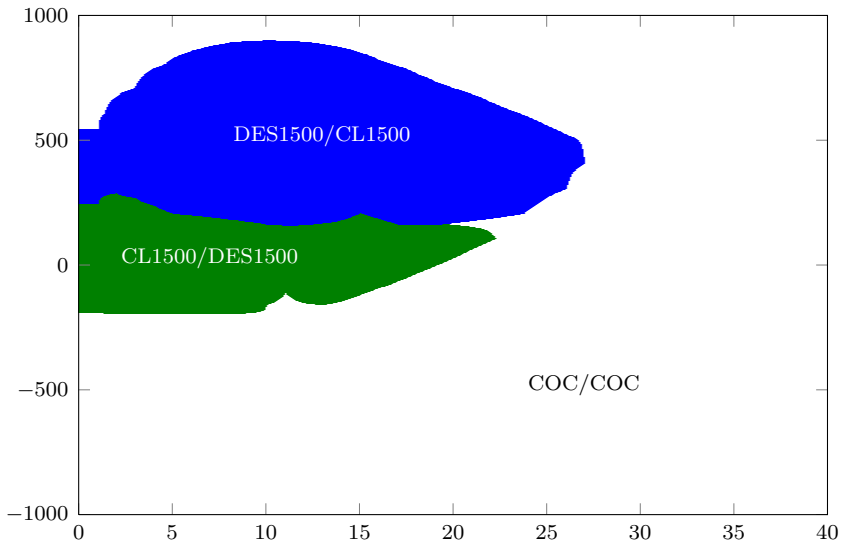
Figure 30 shows the joint policy for two slices of the state space using the synchronous action set, deterministic pilot response, and five terminal cycles. As the figure shows, the alert regions with coordination do not extend as far to the right as the alert regions of the uncoordinated policy (Fig. 20).

Uncoordinated Policy

Instead of using the joint policy, the uncoordinated policy can be used. The assumption in the uncoordinated policy is that the intruder never follows any advisories, which is the assumption the current TCAS logic uses. However, instead of using linear projection like TCAS, the MDP uncoordinated formulation models the intruder following a white-noise acceleration model. The uncoordinated policy will tend to alert earlier and more often than the joint policy because it does not model the increased safety benefit due to the intruder responding to advisories.



(a) $\dot{h}_0 = 0\,\text{ft/min}, \dot{h}_1 = 0\,\text{ft/min}, s_{\text{RA}} = \text{COC/COC}$



(b) $\dot{h}_0 = 1500\,\text{ft/min}, \dot{h}_1 = 0\,\text{ft/min}, s_{\text{RA}} = \text{COC/COC}$

Figure 30. Optimal action plots. Horizontal axis indicates τ (s) and the vertical axis indicates h (ft). Joint action indicates own aircraft action followed by intruder action.

If both aircraft use the uncoordinated policy, most of the time they will issue compatible advisories when there is no sensor noise. However, it is easy to construct situations where, in the absence of communication, the aircraft issue incompatible advisories even with perfect state information. For example, if the aircraft are co-altitude, level, and neither system has issued an advisory, both aircraft will be in the same state and, consequently, issue identical advisories. When there is sensor noise, the aircraft can have different views of the world, resulting in the issuance of more incompatible advisories. Communicating intended advisories between aircraft during execution can help resolve this issue.

Communication Policy

The problem with the uncoordinated policy is that it does not account for the other aircraft following their advisories. However, the uncoordinated model can be augmented to include the joint advisory state. During execution, updating the posterior distribution over the joint advisory state would involve taking into account the advisory being executed by the intruder as inferred from the coordination message. The coordination scheme currently used by TCAS does not allow the exact advisory of the intruder to be communicated, only the complement of the sense employed by the aircraft. However, the VRC can be used to infer a distribution over the intruder advisories and, consequently, the posterior over the joint advisory state.

Introducing the joint advisory state as a state variable in the MDP requires defining its dynamics. One way to do this is to assume that the intruder will continue executing its active advisory indefinitely. Although, in reality, the intruder may reverse, strengthen, or discontinue the advisory, modeling the intruder advisory as constant may still result in a sensible policy. Usually, advisories do not have to be reversed. If the intruder strengthens, it will only result in greater separation between the aircraft, so modeling the intruder as performing a less severe maneuver will only result in more conservative behavior. If the intruder discontinues the advisory, it is usually because the probability of collision at that point is negligible.

Assuming the intruder continues executing the same action indefinitely makes specifying the model very simple. It requires defining $\Pr(s'_{\text{RA}} \mid s_{\text{RA}}, a)$, where s'_{RA} is the joint advisory state at the next step, s_{RA} is the current joint advisory state, and a is the action executed by the own aircraft. If it is important to model changes in the advisory of the intruder, it can be done by introducing a dependence on the other state variables: h, $\dot{h}_0$, $\dot{h}_1$, and τ. These other state variables influence the action taken by the intruder and, consequently, s_{RA} at the next step. It is not expected that the optimized policy will be overly sensitive to the action model of the intruder. The action model could be based on the uncoordinated policy, the joint policy, or the policy of any system (e.g., TCAS) with which the system must interoperate.

8.3 STATE ESTIMATION

State estimation is an important issue in encounters where both aircraft are equipped with a collision avoidance system. If the two aircraft have inconsistent views of the world due to sensor

noise and they do not coordinate their maneuvers, it can lead to an induced collision. This section discusses several ways of performing state estimation.

Independent State Estimation

Different aircraft receive separate sensor measurements, and noise in these measurements results in different state estimates. Most of the time, the belief states between aircraft are relatively consistent. However, there is a risk that the two aircraft in the encounter will have different and incompatible views of the world. For example, both aircraft may believe that they are at the higher altitude, and both issue climb advisories, potentially inducing collision.

TCAS uses independent state estimation, but when there is disagreement between aircraft that results in the aircraft choosing the same sense, it uses a tie-breaking scheme. Each aircraft is assigned a unique 24 bit ICAO (Mode S) address, which is broadcast over the communication channel. If an aircraft encounters another aircraft with a lower ICAO address that selects the same sense, it will reverse its sense. Choosing the exact advisory that is compatible with the coordinated sense is done using individual sensor information. A similar strategy can be employed with logic generated by DP.

Fused State Estimation

If communication bandwidth were not an issue, aircraft could broadcast their measurements. Each aircraft would then use all of the measurements to update its individual state estimates, providing a unified view of the airspace. Unfortunately, the current bandwidth allocated for coordination is insufficient for transmitting accurate measurements. There is also the risk of measurements being corrupted in transit. Although fused state estimation is not feasible, it is worth mentioning because, in theory, it is the best possible strategy with perfect communication.

8.4 ACTION SELECTION

During execution, the various policies computed offline (Section 8.2) can be adapted to account for coordination messages between aircraft.

Independent Action Selection

The simplest strategy is to simply execute the policy directly using the current state estimate without any regard for the actual advisories being executed by the other aircraft. As mentioned earlier, this can lead to induced collisions if the aircraft have inconsistent views of the world or if the uncoordinated policy is used. To assess the value of coordination, the experiments in this section include independent action selection strategies.

Centralized Action Selection

One way to ensure the selection of compatible actions is to have the aircraft with the lower ICAO address dictate the actions for both aircraft. However, adopting this strategy would require changing

the semantics of the current TCAS coordination messages, which only communicate the advisory sense complements, not actual advisories. Another weakness of this strategy is that it uses only the measurements obtained by the aircraft with the lower ICAO address to make decisions, but aircraft are able to measure their own altitude and vertical rate with greater accuracy.

A centralized strategy requires aircraft with higher addresses to trust those with lower addresses. If the sensors on the aircraft with the lower address are faulty or if any other components of the collision avoidance system are not working properly, the action transmitted to the other aircraft can be disastrous. One way to help mitigate such situations is for the aircraft with the higher address to sanity-check the action it receives. If the cost of the prescribed action is significantly higher than the cost of other actions, then the aircraft should execute the lowest cost action. Selecting an action that is contrary to the one assumed by the other aircraft is very risky; it should only be done in extraordinary situations when the prescribed action is anticipated to lead to collision.

Compatibility Action Selection

Another strategy is to force the aircraft with the higher ICAO address to select a compatible action. If there are multiple compatible actions, then the one with the lowest expected cost is selected. Of course, the expected costs are computed assuming that future actions are unimpeded by compatibility restrictions dictated by another aircraft with a potentially different state estimate. Although the expected costs may be inaccurate, performance is not expected to be significantly impaired. The current TCAS logic uses forced compatibility, although it does not choose compatible actions based on expected cost.

8.5 EXAMPLE ENCOUNTER

Figure 31 shows an example encounter simulated from the correlated encounter model. Both aircraft are equipped with the TCAS-like sensor. The performance of TCAS is compared with two coordination strategies. The first strategy, Joint-Indep, consists of using the joint policy with independent action selection based on independent state estimation. Each aircraft maintains a belief state inferred from its own observations, computes the joint action, and selects its portion of the joint action. The second strategy, Joint-Cent, consists of using the joint policy with centralized action selection based on independent state estimation. The aircraft with the lower ICAO address computes the joint action using the belief state inferred from its own observations, selects its portion of the joint action, and transmits the other portion to the other aircraft. In Fig. 31(a), the aircraft approaching from the left has the lower ICAO address. It will be referred to as Aircraft 1; the other aircraft will be referred to as Aircraft 2.

Because the aircraft can disagree about the belief state, the Joint-Indep strategy can potentially cause the aircraft to issue incompatible advisories. Indeed, at $t = 24\,\text{s}$ both aircraft receive descend advisories because they both believe they are at the higher altitude. An NMAC occurs at $t = 40\,\text{s}$; the aircraft come within 15 ft vertically and 356 ft horizontally. In the Joint-Cent strategy, however, because the action selection is centralized, the maneuvers are compatible. At $t = 23\,\text{s}$,

Aircraft 1 issues a descend advisory and transmits a climb advisory to Aircraft 2, which receives it one second later. At the next time step, both advisories are strengthened. The advisories are weakened nine seconds later. The aircraft safely pass each other. TCAS, similarly, selects the same senses: TCAS commands Aircraft 1 to descend at $t = 30\,\text{s}$ and Aircraft 2 to climb at $t = 31\,\text{s}$. The aircraft come within 87 ft vertically and 316 ft horizontally.

8.6 SIMULATION RESULTS

The following three coordination strategies were evaluated in the absence of sensor noise on 500,000 encounters from the correlated encounter model:

- joint policy / independent action selection (Joint-Indep),
- uncoordinated policy / independent action selection (Uncoord-Indep), and
- uncoordinated policy / compatibility action selection (Uncoord-Compat).

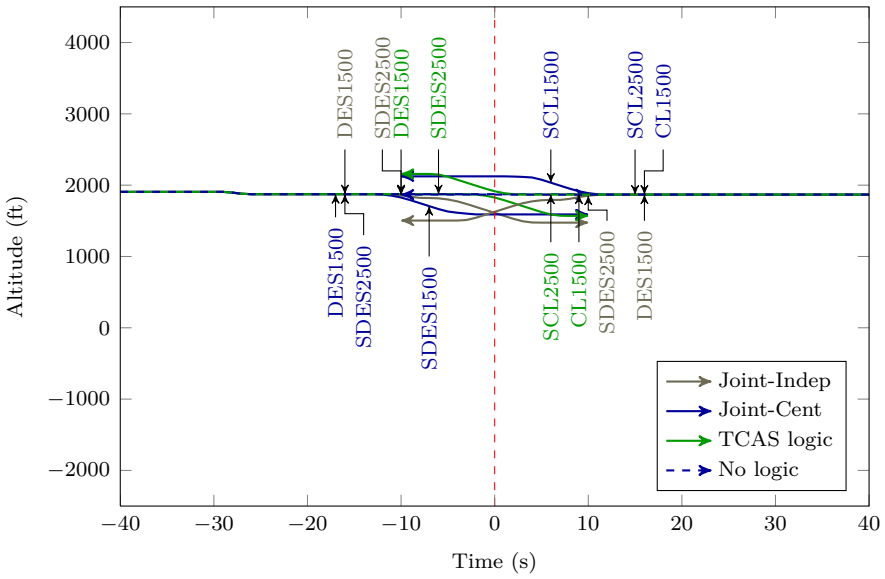
The policies were generated using the same model parameters and costs from Section 5 with the addition of five terminal cycles and no clear-of-conflict reward. The entry distribution is identical to the one used in Section 6 and Section 7. The results are summarized in Table 11, which shows the probability of NMAC and other alert metrics for both aircraft (denoted AC1 and AC2).

TABLE 11

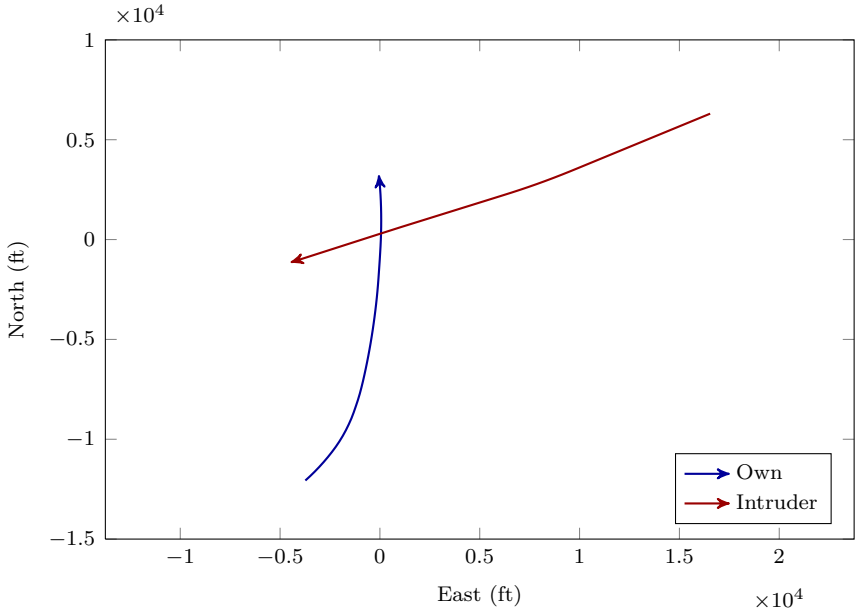
Performance evaluation of coordination strategies with no noise

	Joint-Indep	Uncoord-Indep	Uncoord-Compat	TCAS
Pr(NMAC)	$7.67 \cdot 10^{-6}$	$6.97 \cdot 10^{-6}$	$2.70 \cdot 10^{-6}$	$1.20 \cdot 10^{-5}$
Pr(AC1 alert)	$1.70 \cdot 10^{-1}$	$2.50 \cdot 10^{-1}$	$2.50 \cdot 10^{-1}$	$4.77 \cdot 10^{-1}$
Pr(AC2 alert)	$1.70 \cdot 10^{-1}$	$2.50 \cdot 10^{-1}$	$2.50 \cdot 10^{-1}$	$4.74 \cdot 10^{-1}$
Pr(AC1 strengthening)	$5.60 \cdot 10^{-3}$	$5.90 \cdot 10^{-3}$	$7.90 \cdot 10^{-3}$	$6.78 \cdot 10^{-3}$
Pr(AC2 strengthening)	$5.60 \cdot 10^{-3}$	$5.80 \cdot 10^{-3}$	$6.80 \cdot 10^{-3}$	$7.74 \cdot 10^{-3}$
Pr(AC1 reversal)	$1.94 \cdot 10^{-5}$	$5.04 \cdot 10^{-4}$	$2.51 \cdot 10^{-5}$	$3.91 \cdot 10^{-3}$
Pr(AC2 reversal)	$1.94 \cdot 10^{-5}$	$6.03 \cdot 10^{-4}$	$1.67 \cdot 10^{-4}$	$9.17 \cdot 10^{-3}$
E[AC1 RA changes]	$1.80 \cdot 10^{-1}$	$2.57 \cdot 10^{-1}$	$2.56 \cdot 10^{-1}$	$6.48 \cdot 10^{-1}$
E[AC2 RA changes]	$1.80 \cdot 10^{-1}$	$2.59 \cdot 10^{-1}$	$2.62 \cdot 10^{-1}$	$6.56 \cdot 10^{-1}$
E[AC1 RA duration]	$2.30 \cdot 10^{0}$	$4.59 \cdot 10^{0}$	$4.60 \cdot 10^{0}$	$1.08 \cdot 10^{1}$
E[AC2 RA duration]	$2.30 \cdot 10^{0}$	$4.64 \cdot 10^{0}$	$4.60 \cdot 10^{0}$	$1.07 \cdot 10^{1}$

The Joint-Indep strategy outperforms TCAS on all metrics: Joint-Indep is almost twice as safe and alerts almost three times less frequently. Because there is no sensor noise, Uncoord-Indep is just as safe as Joint-Indep but at the expense of more alerts. The Uncoord-Compat strategy is over twice as safe as Uncoord-Indep with the same alert rate. This is because Uncoord-Compat, unlike Uncoord-Indep, ensures compatibility. Uncoord-Compat is over four times safer than TCAS while alerting almost half the time.



(a) Vertical profile.



(b) Horizontal profile.

Figure 31. Example coordinated encounter comparing the performance of DP coordination strategies with TCAS.

The following four coordination strategies were evaluated with sensor noise (Section 7.1) on the same encounters selected from the correlated encounter model:

- joint policy / independent action selection (Joint-Indep),
- uncoordinated policy / independent action selection (Uncoord-Indep),
- joint policy / centralized action selection (Joint-Cent), and
- uncoordinated policy / compatibility action selection (Uncoord-Compat).

The results are summarized in Table 12.

TABLE 12

Performance evaluation of coordination strategies with the TCAS sensor

	Joint-Indep	Uncoord-Indep	Joint-Cent	Uncoord-Compat	TCAS
Pr(NMAC)	$1.65 \cdot 10^{-4}$	$3.04 \cdot 10^{-5}$	$1.51 \cdot 10^{-5}$	$6.77 \cdot 10^{-6}$	$1.37 \cdot 10^{-5}$
Pr(AC1 alert)	$2.97 \cdot 10^{-1}$	$4.07 \cdot 10^{-1}$	$3.05 \cdot 10^{-1}$	$4.07 \cdot 10^{-1}$	$5.12 \cdot 10^{-1}$
Pr(AC2 alert)	$2.93 \cdot 10^{-1}$	$4.10 \cdot 10^{-1}$	$3.04 \cdot 10^{-1}$	$4.10 \cdot 10^{-1}$	$5.09 \cdot 10^{-1}$
Pr(AC1 strengthening)	$1.74 \cdot 10^{-2}$	$1.92 \cdot 10^{-2}$	$1.84 \cdot 10^{-2}$	$1.87 \cdot 10^{-2}$	$6.54 \cdot 10^{-3}$
Pr(AC2 strengthening)	$1.48 \cdot 10^{-2}$	$1.76 \cdot 10^{-2}$	$1.84 \cdot 10^{-2}$	$2.00 \cdot 10^{-2}$	$8.01 \cdot 10^{-3}$
Pr(AC1 reversal)	$8.99 \cdot 10^{-4}$	$5.00 \cdot 10^{-3}$	$1.53 \cdot 10^{-4}$	$1.40 \cdot 10^{-3}$	$3.60 \cdot 10^{-3}$
Pr(AC2 reversal)	$6.92 \cdot 10^{-4}$	$3.90 \cdot 10^{-3}$	$1.53 \cdot 10^{-4}$	$1.90 \cdot 10^{-3}$	$7.16 \cdot 10^{-3}$
E[AC1 RA changes]	$3.27 \cdot 10^{-1}$	$4.31 \cdot 10^{-1}$	$3.36 \cdot 10^{-1}$	$4.26 \cdot 10^{-1}$	$6.94 \cdot 10^{-1}$
E[AC2 RA changes]	$3.19 \cdot 10^{-1}$	$4.28 \cdot 10^{-1}$	$3.41 \cdot 10^{-1}$	$4.36 \cdot 10^{-1}$	$7.07 \cdot 10^{-1}$
E[AC1 RA duration]	$4.73 \cdot 10^{0}$	$8.44 \cdot 10^{0}$	$4.84 \cdot 10^{0}$	$8.48 \cdot 10^{0}$	$1.13 \cdot 10^{1}$
E[AC2 RA duration]	$4.57 \cdot 10^{0}$	$8.38 \cdot 10^{0}$	$4.83 \cdot 10^{0}$	$8.41 \cdot 10^{0}$	$1.13 \cdot 10^{1}$

The Joint-Indep strategy performs markedly worse than TCAS. Because the aircraft can disagree over the belief state, they can issue incompatible advisories that if followed lead to collision. The Uncoor-Indep strategy, though it also has the potential of issuing incompatible advisories, is safer than Joint-Indep because it alerts earlier, perhaps allowing additional time to reverse the advisory if necessary. Joint-Cent, which solves this problem in part by using only the belief state of the aircraft with the lower address, has an NMAC probability ten times lower than Joint-Indep with comparable alert probability. The Joint-Cent strategy still has a somewhat greater probability of NMAC compared to TCAS but alerts significantly less. The Uncoord-Compat strategy is over twice as safe as TCAS and alerts less often. One potential reason why Uncoord-Compat is safer than Joint-Cent is because Uncoord-Compat, using the uncoordinated logic which assumes that intruders do not maneuver to avoid collision, alerts more aggressively.

8.7 DISCUSSION

This section discussed numerous coordination strategies and presented some results indicating that new coordination techniques based on solving MDPs can exceed the level of safety provided by

the current TCAS logic with less disruption to the pilots. This section demonstrated that, in the presence of realistic sensor noise, the Joint-Cent and Uncoord-Compat strategies are viable methods of achieving coordination. Future work will examine the use of incorporating communication into the model. This section only considered evaluation of the joint MDP using the synchronous action set. Future work could reveal the extent to which strategies that use the solution of the joint MDP can be improved through the use of different joint action spaces.

The Uncoord-Compat strategy may be the most appealing way of providing coordination because it can preserve the current communication mechanism of TCAS. The future system would transmit VRC messages potentially inhibiting the advisories from which the intruder could select. An intruder equipped with the same system or with any prior version of the TCAS logic would be able to process the coordination message. Though the Uncoord-Compat strategy does not represent the optimal solution to the problem of coordination, it has been shown to perform remarkably well in realistic encounter situations. It does not require offline planning over the joint MDP; no additional cost table beyond that for the uncoordinated MDP would have to be loaded on board the aircraft.

9. MULTIPLE THREAT ENCOUNTERS

The previous sections of this report have focused entirely on resolving encounters with single intruders. Although encounters involving two or more intruders are relatively rare in the current airspace, the collision avoidance logic must be robust to such situations. This section generalizes the methods introduced earlier in this report to multithreat encounters.

Ideally, a solution would be obtained from an MDP modeling an arbitrary number of aircraft. Unfortunately, each additional intruder would require introducing at least two variables to capture their relative altitude and vertical rate. Intruders with collision avoidance systems would require additional advisory state variables. Augmenting the MDP in Section 3 would require at least two orders of magnitude more processing time to compute the optimal cost table. Storage of the table would require at least two orders of magnitude more memory, which is not currently feasible.

One strategy for handling large MDPs is to decompose the decision making to multiple agents [50]. In the collision avoidance problem, each agent may be assigned a particular intruder to avoid. They independently make recommendations whether to alert and which advisory to issue, and some strategy is employed to make the best overall decision. Several different kinds of strategies have been proposed, including command arbitration and utility fusion. TCAS uses a method that resembles command arbitration, but the MDP approach permits utility fusion, which may be more appropriate for resolving multithreat encounters.

9.1 COMMAND ARBITRATION

In command arbitration, an arbiter selects one of the actions recommended by one of the agents. One of the simplest command arbitration strategies is to accept the recommendation of the agent whose intruder is closest. In multithreat encounters, one intruder is usually much closer than the others, and so resolving NMACs sequentially in this manner may be satisfactory most of the time. However, it is easy to construct situations where this strategy fails.

Figure 32 shows the action regions for the closest command arbitration strategy for an encounter with two approaching intruders that are equally distant from the own aircraft horizontally and are separated from each other by 400 ft. All aircraft are currently level, and no advisory has been issued. The actions taken by this strategy appear sensible for much of the state space. However, it does especially poorly when the own aircraft is between the two intruders in altitude. When the own aircraft is in the position indicated in the figure, the strategy results in a descend advisory because the closer intruder is above. This descend advisory, though, results in the aircraft descending into the other intruder. Of course, the system would later reverse the advisory to prevent collision, but this behavior is not ideal. Issuing a climb advisory or even delaying the alert would be preferable.

TCAS adopts a more sophisticated command arbitration strategy. It treats each threat individually, with the same threat detection, initial sense selection, and initial strength selection logic that would be used with a single intruder. The multithreat portion of the logic attempts to reconcile the senses and strengths associated with each intruder before displaying a composite

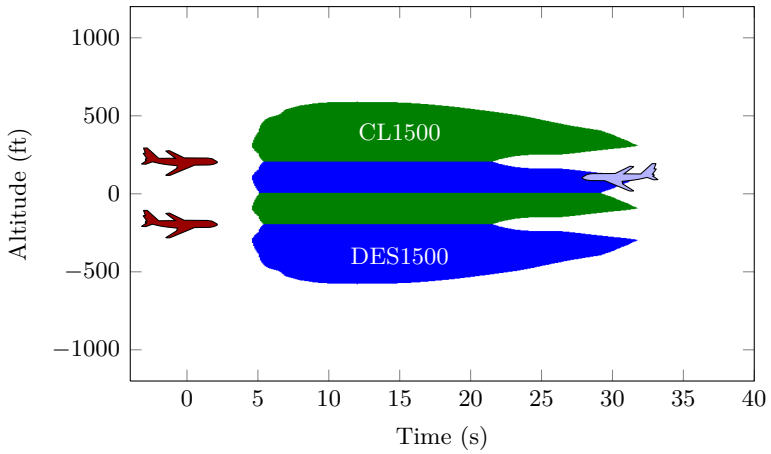


Figure 32. Closest command arbitration strategy. Two intruder aircraft are on the left, and the own aircraft is on the right. All aircraft are initially level and no advisory has been issued. The parameters from Section 3 with zero terminal cycles were used.

advisory to the pilots. When all threats have the same sense, the logic simply uses the individual advisory with the greatest strength. When the senses of the individual advisories differ, TCAS uses a set of rules to either (1) identify a single sense for all threats or (2) issue a "dual-negative advisory" that places vertical speed limits in both senses.

9.2 UTILITY FUSION

Instead of making decisions based solely on the actions recommended by the agents, the utility (or, alternatively, expected cost) each agent assigns to each action can be leveraged to arrive at a better overall action [51]. Let $J^{(n)}(a)$ be the expected cost for executing action a as assigned by agent n. This value can be interpreted as the expected cost when executing action a for one step and then continuing with the optimal pairwise individual logic, ignoring all intruders other than intruder n.

Fusing these action costs requires defining a function f that combines the costs from all of the agents. The action to be executed is given by

$$\arg\min_a f(J^{(1)}(a), \ldots, J^{(N)}(a)). \tag{32}$$

There are different ways to define the combining function f, but it should have the following properties:

- *Exchangeability:* $f(\ldots, a, \ldots, b, \ldots) = f(\ldots, b, \ldots, a, \ldots)$ for all a and b.
- *Monotonicity:* $f(\ldots, a, \ldots) \leq f(\ldots, b, \ldots)$ for all $a < b$.
- *One-step optimality:* The recommended action should be optimal for a one-step horizon.

One way to define f is simply as a summation [50]. However, it does not seem appropriate to sum the immediate costs in this problem. For example, the penalty for alerting in this scheme would be incurred for each intruder. Double counting alert costs results in delaying the alert, which is especially undesirable in multithreat encounters. Depending on the encounter situation, it may be important to alert earlier. Figure 33(a) shows a slice of the policy for this strategy.

Alternatively, the combining function can return the maximum expected cost. At each time step, the strategy selects the action that guarantees the lowest expected cost against all individual intruders in isolation. One advantage of this approach is that it does not double count the alert costs. As shown in Fig. 33(b), this guaranteed cost strategy results in an alerting region similar to the closest command arbitration strategy, but it does not have the issue identified earlier where the own aircraft descends to avoid the closer intruder but results in entering the path of another intruder.

The *sum-collision/max-action* (SCMA) strategy involves partitioning the expected cost due to collision from the expected cost due to alerting.[4] Given an agent n and action a,

$$J^{(n)}(a) = J^{(n)}_{\text{collision}}(a) + J^{(n)}_{\text{action}}(a). \tag{33}$$

DP can be used offline to compute the collision cost and the action cost for every discrete state-action pair. The results would be stored in a table. During execution, f would combine these costs as follows:

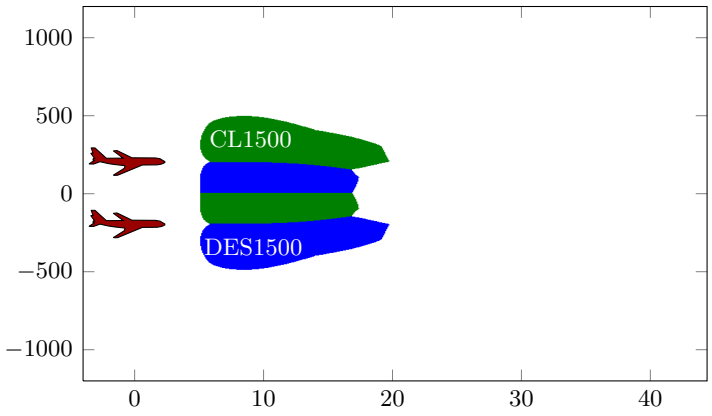
$$f(J^{(1)}_{\text{collision}}(a), J^{(1)}_{\text{action}}(a), \ldots, J^{(N)}_{\text{collision}}(a), J^{(N)}_{\text{action}}(a)) = \sum_n J^{(n)}_{\text{collision}}(a) + \max_n J^{(n)}_{\text{action}}(a). \tag{34}$$

Figure 33(c) shows the alert regions for SCMA. It does not double count alert costs, but it does sum the collision costs. Because collision with two aircraft is twice as costly as collision with one aircraft, the alert region is enlarged.

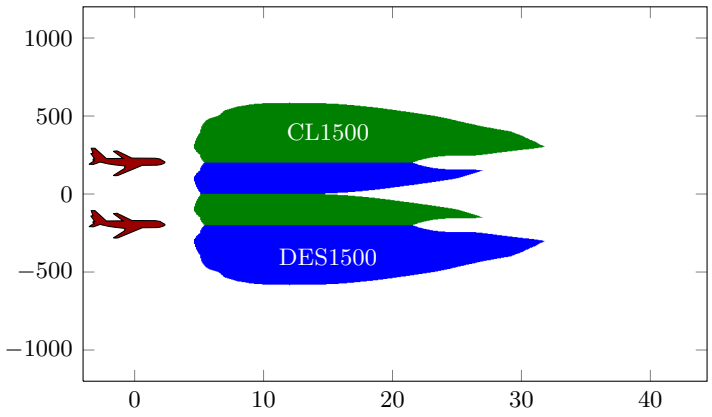
9.3 EXAMPLE ENCOUNTER

Figure 34 exhibits the behavior of the guaranteed cost strategy and the TCAS logic in an example multithreat encounter. The multithreat encounter was sampled from a multithreat model built from radar data [52]. There is no sensor noise, and both intruders are unequipped. Eleven seconds into the encounter, the DP logic issues DES1500. The expected costs for issuing DES1500, CL1500, and COC are 0.00595, 0.00741, and 0.00601 for intruder 1, respectively. As intruder 2 is moving away from the own aircraft, thus posing no significant threat, the expected costs are 0.001, 0.001, and 0 for intruder 2, respectively. Fourteen seconds later, the DP logic issues SDES2500. As the advisory is later terminated, the own aircraft levels off, safely passing between both intruders in altitude. One potential issue with the guaranteed cost strategy is that it can cause the own aircraft to pass between the intruders, as shown in Fig. 33(b). If intruder 2 happened to be a more imminent threat, the own aircraft would have had to climb after its descent. The TCAS logic issues CL1500 and subsequently SCL2500 to pass over both intruders, but they are too late. An NMAC occurs with intruder 1.

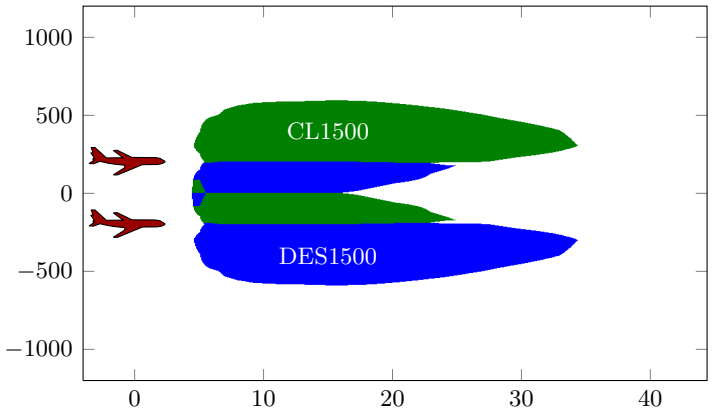
[4] The SCMA strategy was originally proposed by Richard Williams from Lincoln Laboratory.



(a) Summation utility fusion strategy.

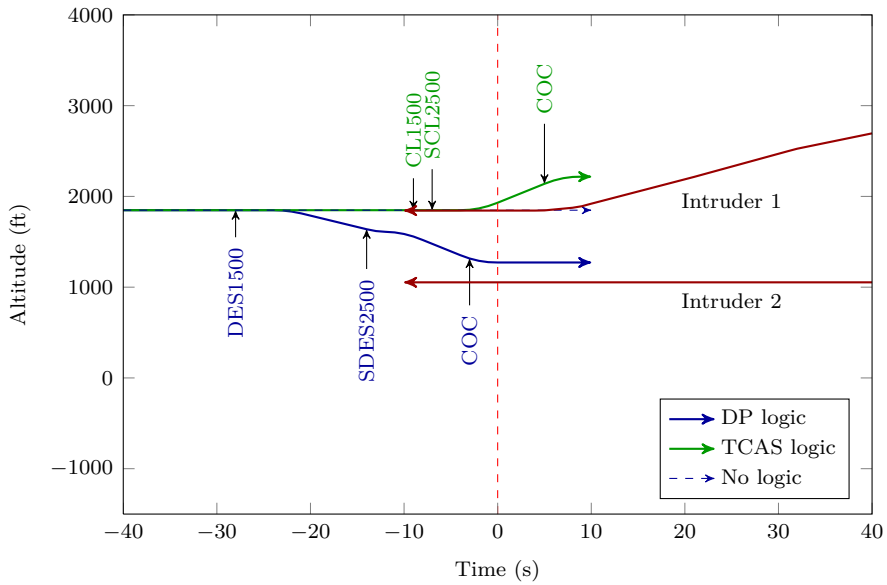


(b) Guaranteed cost strategy.

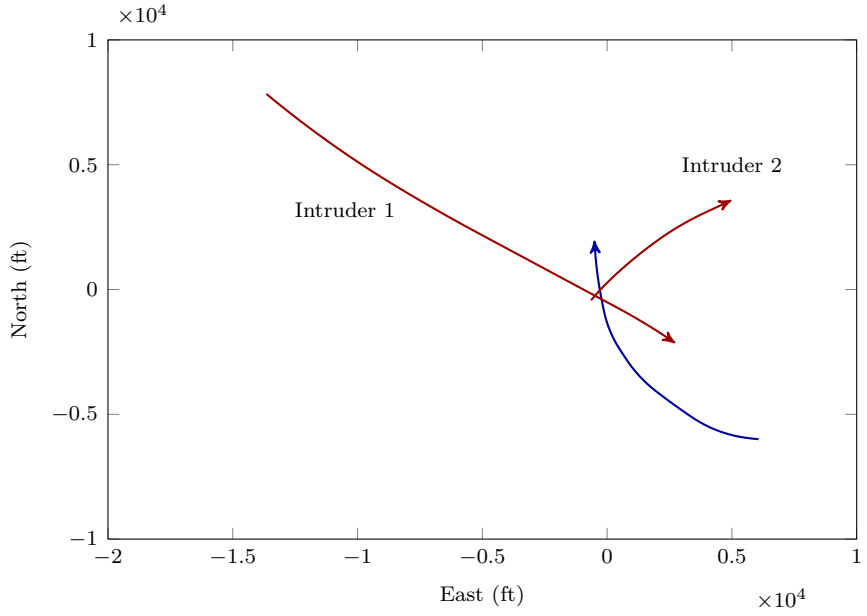


(c) SCMA strategy.

Figure 33. Utility fusion strategies. Two intruder aircraft are on the left. All aircraft are initially level and no advisory has been issued. Horizontal axis is time (s) and the vertical axis is altitude (ft). The parameters from Section 3 with zero terminal cycles were used.



(a) Vertical profile.



(b) Horizontal profile.

Figure 34. Example multithreat encounter comparing the performance of a DP multithreat strategy with TCAS.

9.4 SIMULATION RESULTS

Table 13 and Table 14 summarize the results of evaluating the guaranteed cost and summation utility fusion strategies on 500,000 multithreat encounters. In these encounters one aircraft, equipped with the collision avoidance system, encounters two unequipped intruders. The tables summarize the performance of the strategies in the absence of sensor noise and when the equipped aircraft uses the TCAS-like sensor. The performance of the TCAS multithreat logic is also shown.

In Table 13, the DP logic was optimized using a clear-of-conflict reward of 1×10^{-4}. With the exception of the clear-of-conflict reward, all the other parameters used to generate the logic were identical to those used in Section 8. In both the perfect sensing and TCAS sensing scenarios, the DP strategies resolve more NMACs than TCAS; they also issue fewer alerts. The summation strategy results in fewer alerts, and consequently more NMACs, than the guaranteed cost strategy because it double counts alert costs.

In Table 14, the DP logic was optimized using no clear-of-conflict reward. This has the effect of making the DP logic safer at the expense of increasing the alert rate and the mean time for which the system displays alerts. The guaranteed cost strategy is almost four times safer than TCAS when there is no sensor noise and twice as safe when there is TCAS-like sensor noise. Removing the clear-of-conflict reward also has the added advantage of generally decreasing the strengthening and reversal rates of the DP logic.

9.5 DISCUSSION

The experiments demonstrate that combining the expected costs for individual threats can lead to effective avoidance behavior that surpasses TCAS in safety with fewer alerts. It appears that this approach has not been pursued in the literature for aircraft collision avoidance. Further analysis is required to identify potential vulnerabilities of this approach. This section focused on encounters where only a single aircraft is equipped with a collision avoidance system. Future work will involve incorporating coordination between equipped aircraft and studying its impact on overall safety.

TABLE 13

Multithreat performance evaluation with clear-of-conflict reward

	Perfect Sensor			TCAS Sensor		
	Guaranteed cost	Summation	TCAS	Guaranteed cost	Summation	TCAS
Pr(NMAC)	$1.14 \cdot 10^{-3}$	$1.54 \cdot 10^{-3}$	$2.94 \cdot 10^{-3}$	$1.99 \cdot 10^{-3}$	$2.32 \cdot 10^{-3}$	$2.96 \cdot 10^{-3}$
Pr(Alert)	$6.07 \cdot 10^{-1}$	$5.46 \cdot 10^{-1}$	$7.47 \cdot 10^{-1}$	$7.12 \cdot 10^{-1}$	$6.38 \cdot 10^{-1}$	$7.64 \cdot 10^{-1}$
Pr(Strengthening)	$1.40 \cdot 10^{-1}$	$1.27 \cdot 10^{-1}$	$4.49 \cdot 10^{-2}$	$2.57 \cdot 10^{-1}$	$1.90 \cdot 10^{-1}$	$4.43 \cdot 10^{-2}$
Pr(Reversal)	$7.35 \cdot 10^{-3}$	$5.41 \cdot 10^{-3}$	$5.50 \cdot 10^{-3}$	$1.63 \cdot 10^{-2}$	$1.06 \cdot 10^{-2}$	$6.55 \cdot 10^{-3}$
E[RA changes]	$1.42 \cdot 10^{0}$	$1.13 \cdot 10^{0}$	$1.25 \cdot 10^{0}$	$1.90 \cdot 10^{0}$	$1.45 \cdot 10^{0}$	$1.30 \cdot 10^{0}$
E[RA duration]	$1.14 \cdot 10^{1}$	$9.10 \cdot 10^{0}$	$1.92 \cdot 10^{1}$	$1.18 \cdot 10^{1}$	$9.80 \cdot 10^{0}$	$1.95 \cdot 10^{1}$

TABLE 14

Multithreat performance evaluation with no clear-of-conflict reward

	Perfect Sensor			TCAS Sensor		
	Guaranteed cost	Summation	TCAS	Guaranteed cost	Summation	TCAS
Pr(NMAC)	$8.52 \cdot 10^{-4}$	$1.06 \cdot 10^{-3}$	$2.94 \cdot 10^{-3}$	$1.42 \cdot 10^{-3}$	$1.68 \cdot 10^{-3}$	$2.96 \cdot 10^{-3}$
Pr(Alert)	$6.22 \cdot 10^{-1}$	$5.55 \cdot 10^{-1}$	$7.47 \cdot 10^{-1}$	$7.28 \cdot 10^{-1}$	$6.49 \cdot 10^{-1}$	$7.64 \cdot 10^{-1}$
Pr(Strengthening)	$4.96 \cdot 10^{-2}$	$3.89 \cdot 10^{-2}$	$4.49 \cdot 10^{-2}$	$9.00 \cdot 10^{-2}$	$6.65 \cdot 10^{-2}$	$4.43 \cdot 10^{-2}$
Pr(Reversal)	$7.37 \cdot 10^{-3}$	$5.40 \cdot 10^{-3}$	$5.46 \cdot 10^{-3}$	$1.60 \cdot 10^{-2}$	$1.03 \cdot 10^{-2}$	$6.55 \cdot 10^{-3}$
E[RA changes]	$7.06 \cdot 10^{-1}$	$6.15 \cdot 10^{-1}$	$1.25 \cdot 10^{0}$	$9.06 \cdot 10^{-1}$	$7.57 \cdot 10^{-1}$	$1.30 \cdot 10^{0}$
E[RA duration]	$1.49 \cdot 10^{1}$	$1.16 \cdot 10^{1}$	$1.92 \cdot 10^{1}$	$1.85 \cdot 10^{1}$	$1.52 \cdot 10^{1}$	$1.95 \cdot 10^{1}$

This page intentionally left blank.

10. CONCLUSIONS

This report has explored the use of dynamic programming (DP) as a method for automatically deriving robust airborne collision avoidance logic. The experiments demonstrate that this approach has the potential to significantly improve safety while reducing the rate of unnecessary alerts compared to the current TCAS logic. In addition, the method satisfies the collection of design considerations introduced at the beginning of the report. This section summarizes how the approach addresses these considerations and identifies areas where further research is required.

- *Safety performance.* The simulations in this report demonstrate that DP can further reduce the risk of near mid-air collision (NMAC) beyond what is currently provided by TCAS while reducing the alert rate. The experiments were conducted using a high-fidelity encounter model derived from nine months of national radar data, and NMAC rates were estimated from millions of simulations. The safety provided by the DP logic is a function of the parameters of the cost function against which the logic is optimized. The NMAC rate can be reduced further at the expense of additional alerts, strengthenings, and reversals. Determining the cost function parameters will be an important area of discussion within the TCAS development community because of their impact on safety and operational performance. Further simulation studies will better inform the choice of cost function parameters.

- *Operational performance.* The experiments demonstrate that the DP logic results in significantly fewer alerts than TCAS while improving safety. In addition, the DP logic issues fewer reversals. For the cost settings used in this report, the DP logic would strengthen its advisories more frequently than TCAS. However, strengthening is a much less severe maneuver than reversing. The cost parameters can be adjusted to reduce the rate of strengthening, but it will be at the expense of additional alerts, additional reversals, or reduced safety. Deciding how to balance these operational considerations will require engaging the TCAS development community.

- *Onboard computation.* Since the cost tables and the entry distributions are computed offline, very little computation is required on the aircraft. The onboard computation primarily involves state estimation, table lookups, and interpolation. Many other approaches to collision avoidance suggested in the literature require substantial online computation, which makes them infeasible for a system that needs to update its decisions at least once per second.

- *Coordination.* This report discussed several coordination strategies to reduce the risk of induced collision with other aircraft equipped with collision avoidance systems. Some of the coordination strategies are compatible with the existing coordination mechanism used by TCAS. Further work will be required to refine these strategies, but they have already been demonstrated to outperform the current version of TCAS.

- *Unequipped aircraft.* Most of this report has focused on encounters with aircraft that are not equipped with a collision avoidance system. It was found that the DP logic significantly outperforms the existing TCAS logic according to both safety and operational metrics.

- *Interoperability.* The introduction of a new collision avoidance system will require, at least initially, the ability to interoperate with legacy systems. Many of the coordination strategies suggested in this report are directly compatible with the current version of TCAS. Further work is required to analyze the performance of the DP logic in encounters with TCAS.

- *Surveillance compatibility.* The DP logic can accommodate different surveillance technologies. The expected cost tables are not dependent on the sensor error model. Sensor error is accounted for during execution. Incorporating a new sensor system would require updating the tracker, which produces a probability distribution over the state space based on sensor measurements. The rest of the logic that averages the expected cost over the state distribution remains the same. Of course, not all surveillance systems provide the same level of accuracy. If a surveillance system is introduced that provides relatively poor state estimates, the cost of alerting should be decreased so that the safety level is not degraded. This report has focused on the current TCAS surveillance system, but a future collision avoidance system is likely to incorporate improved sensors that reduce the error in bearing measurement, leading to additional performance gains.

- *Aircraft performance constraints.* The accommodation of aircraft performance constraints was not a focus of this report, although DP is well suited for incorporation of such constraints. The only performance constraint in the experiments was that the vertical rate be kept within $\pm 2500\,\mathrm{ft/min}$, but that range could have just as easily been something different to reflect the limits of less capable aircraft. One attractive aspect of the way DP handles these constraints is that it does not simply restrict the current space of available actions; it actually chooses the current action with the knowledge that in the future the space of available actions will be restricted. Aircraft with different categories of capabilities would require different cost tables. If aircraft broadcast their performance limits, then these limitations can be taken into account when modeling the intruder aircraft. A collection of different cost tables can be optimized offline for different intruder capabilities, and the appropriate table would be indexed into during flight.

- *Extensibility.* Changing the probabilistic model to better reflect the evolving airspace characteristics and pilot behavior is relatively straightforward, and re-optimizing the logic in response to these changes does not require any human effort. Modifying TCAS pseudocode is much more complicated. The primary limitation of the DP approach is that the model must be Markovian and must be capable of being approximated well by a discrete state space of manageable size. Other DP methods could be employed besides the one used in this report to help scale the method to more complex models.

- *Validation.* The validation of the DP logic will likely involve many of the same steps used to validate TCAS, including Monte Carlo simulation and flight tests. Additional validation techniques can be employed due to the fact that the model used by DP is Markovian. This report showed to efficiently compute the probability of NMAC as well as other performance metrics across the entire state space in under a minute. Performing the same computation using Monte Carlo, which is what would be required for a logic like TCAS, would have required months of continuous computation.

- *Verification.* The correctness of a manufacturer's implementation of the TCAS pseudocode is verified against a set of test encounters. The DP logic can be verified in the same way. However, just because an implementation satisfies all test cases does not mean that it is correct since it is not possible to test every possible encounter situation. With the DP logic, the complexity of the logic can be encoded using numerical lookup tables that can be standardized and delivered to manufacturers. Because there is very little code for manufacturers to implement, there is less opportunity for errors to be introduced.

This report has answered the primary research questions left open in Project Report ATC-360, but additional study of multiple threat encounters and interoperability is still required. Further study will also compare the behavior of the DP logic to other approaches, such as geometric optimization, that do not leverage probabilistic models to infer the optimal course of action. Before committing to a particular development strategy for future TCAS it is important to evaluate the relative merits of different approaches from the perspective of development cost and risk over the entire life cycle of the system, in addition to anticipated safety and operational performance. A future paper will address these issues.

This page intentionally left blank.

APPENDIX A
EXPECTED COST FILE FORMAT

This appendix describes the file format for the expected cost table. The expected cost table contains the expected costs stored using the 64 bit IEEE 754-1985 double precision floating point format. Because the number of available actions from a particular state is small compared to the total number of actions (e.g., one cannot strengthen or reverse until an initial advisory is issued), memory may be more efficiently utilized by only storing the costs for the available actions.

An index file is used to efficiently locate the position in the expected cost table. The index file contains a sequence of unsigned 32 bit integers, containing offsets into the cost table. For example, in Fig. A-1, the expected costs for the second state can be found by skipping the first three entries in the table, corresponding to 3×64 bits from the beginning of the file. If more than $2^{32} - 1$ state-action pairs are needed, then the representation of the index file would need to be increased from 32 bits. If there are N states, then there are $N + 1$ entries in the index file. The last entry contains the offset to one entry past the last.

For each state, the cost table contains a sequence of values. To determine which value corresponds to which action, a table of actions is used. Because there are no more than 255 actions, the action file uses unsigned 8 bit numbers. The action file contains the same number of entries as the cost table, but it is eight times smaller because it uses an 8 bit instead of a 64 bit representation.

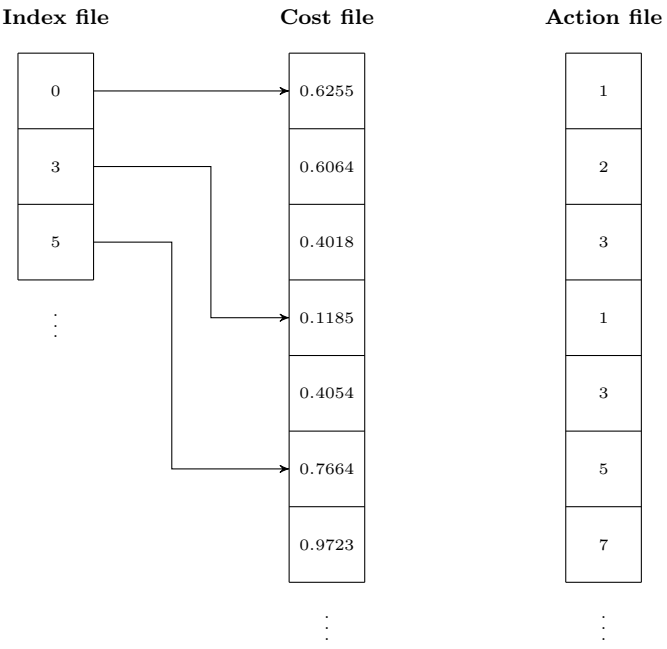


Figure A-1. Notional diagram of expected cost file format.

This page intentionally left blank.

APPENDIX B
TRACKING

This appendix reviews the Kalman filter and unscented Kalman filter and discusses the horizontal and vertical motion trackers used in the evaluation of the dynamic programming logic in Section 7.

B.1 KALMAN FILTER

The Kalman filter is a minimum mean square error (MMSE) estimator that is optimal if the initial state of the system as well as all the noise entering the system are Gaussian. Otherwise, the Kalman filter is the best estimator among the class of linear MMSE state estimators. This section briefly derives the Kalman filter equations. The following discussion is adapted from [44], pp. 200–207.

Consider the following discrete-time linear dynamic system:

$$x(k+1) = F(k)x(k) + G(k)u(k) + B(k)v(k), \tag{B-1}$$

where $x(k) \in \mathbb{R}^n$ is the state vector at time k, $u(k)$ is a known input (i.e., control) vector, and $v(k)$ is zero-mean white Gaussian process noise with covariance $Q(k)$. The matrix $F(k)$ is the (possibly) time-varying system dynamics matrix, $G(k)$ is the input gain matrix, and $B(k)$ is the state-noise coupling matrix. Additionally, any measurements $z(k) \in \mathbb{R}^m$ of the system are modeled by

$$z(k) = H(k)x(k) + w(k), \tag{B-2}$$

where $w(k)$ is zero-mean white Gaussian measurement noise with covariance $R(k)$, and $H(k)$ is the measurement matrix describing the transformation from the state space in $\mathbb{R}^n$ to the measurement space in $\mathbb{R}^m$. Let the initial state $x(0)$ be modeled as a Gaussian random variable with known mean and covariance, and let the noise sequences $v(k)$ and $w(k)$ be mutually independent.

It can be shown that estimators that minimize the mean square error (the difference between the true value of the state and the estimate) have as their optimal estimate at every time k the conditional mean of the state at time k given the measurements up to and including time k. That is,

$$\hat{x}(k \mid k) \equiv \mathrm{E}[x(k) \mid Z^k], \tag{B-3}$$

where $\hat{x}(k \mid k)$ denotes the state estimate at time k given all information up to and including time k (i.e., a posterior state estimate) and Z^k is the sequence of measurements up to and including time k. For jointly distributed vectors x and z, the conditional mean is given by

$$\hat{x} = \mathrm{E}[x \mid z] = \bar{x} + P_{xz}P_{zz}^{-1}(z - \bar{z}) \tag{B-4}$$

and the covariance by

$$P_{xx|z} = \mathrm{E}[(x - \hat{x})(x - \hat{x})^T \mid z] = P_{xx} - P_{xz}P_{zz}^{-1}P_{zx}, \tag{B-5}$$

where $\bar{x}$ is the mean of x, P_{xx} is the covariance of x, P_{zz} is the covariance of z, and P_{xz} $(= P_{zx}^T)$ is the cross-covariance between x and z.

In the context of the Kalman filter, x and z would be the state and measurement vectors, respectively. The conditional mean $\hat{x}$ of Eq. (B-4) would represent the updated state estimate at time $k+1$, $\hat{x}(k+1 \mid k+1)$, and the mean $\bar{x}$ would represent the one-step predicted state at time $k+1$ from time k, $\hat{x}(k+1 \mid k)$. Moreover, z would be the actual measurement at time $k+1$, $z(k+1)$, and $\bar{z}$, the predicted measurement at time $k+1$, $\hat{z}(k+1 \mid k)$. The product $P_{xz}P_{zz}^{-1}$, often called the filter gain, would then be a measure of the amount of correction that would have to be applied to $\hat{x}(k+1 \mid k)$ due to the measurement prediction error $z(k+1) - \hat{z}(k+1 \mid k)$.

The Kalman filter fundamentally consists of two steps: a predict step and an update step. In the predict step, the state estimate at time $k+1$, $\hat{x}(k+1 \mid k)$, is predicted based on the posterior state estimate at time k, $\hat{x}(k \mid k)$:

$$\hat{x}(k+1 \mid k) = \mathrm{E}[x(k+1) \mid Z^k] = F(k)\hat{x}(k \mid k) + G(k)u(k), \tag{B-6}$$

where the term involving the noise drops out because $\mathrm{E}[v(k)] = 0$ by definition. Note that in the prediction the measurement at time $k+1$ is not used. The state prediction covariance simplifies to

$$P(k+1 \mid k) = F(k)P(k \mid k)F(k)^T + B(k)Q(k)B(k)^T, \tag{B-7}$$

where $P(k \mid k)$ is the covariance of the posterior state estimate. One could predict what the measurement at time $k+1$ would be by simply transforming the predicted state estimate to arrive at

$$\hat{z}(k+1 \mid k) = \mathrm{E}[z(k+1) \mid Z^k] = H(k+1)\hat{x}(k+1 \mid k), \tag{B-8}$$

where the zero-mean noise is again not present. Similar to Eq. (B-7), the measurement prediction covariance is given by

$$S(k+1) = H(k+1)P(k+1 \mid k)H(k+1)^T + R(k+1), \tag{B-9}$$

which is equivalent to P_{zz} described above. Calculating the cross-covariance between the state and measurement, P_{xz}, one arrives at the equation for the filter gain of the Kalman filter:

$$W(k+1) = P(k+1 \mid k)H(k+1)^T S(k+1)^{-1}. \tag{B-10}$$

In the update step, the predicted state and covariance are updated using the measurement. Using Eq. (B-4), the updated state estimate at time $k+1$ is given by

$$\hat{x}(k+1 \mid k+1) = \hat{x}(k+1 \mid k) + W(k+1)\nu(k+1), \tag{B-11}$$

where $\nu(k+1) \equiv z(k+1) - \hat{z}(k+1 \mid k)$ is the measurement residual or innovation. The quantity S can be viewed as the innovation covariance. The updated covariance is obtained by using Eq. (B-5) and making the necessary substitutions:

$$P(k+1 \mid k+1) = P(k+1 \mid k) - W(k+1)S(k+1)W(k+1)^T. \tag{B-12}$$

Figure B-1 is a high-level overview representing one cycle of the Kalman filter that summarizes the relevent equations presented above. Note that the covariance is independent of the state and measurements and can be calculated offline, as Eq. (B-12) implies.

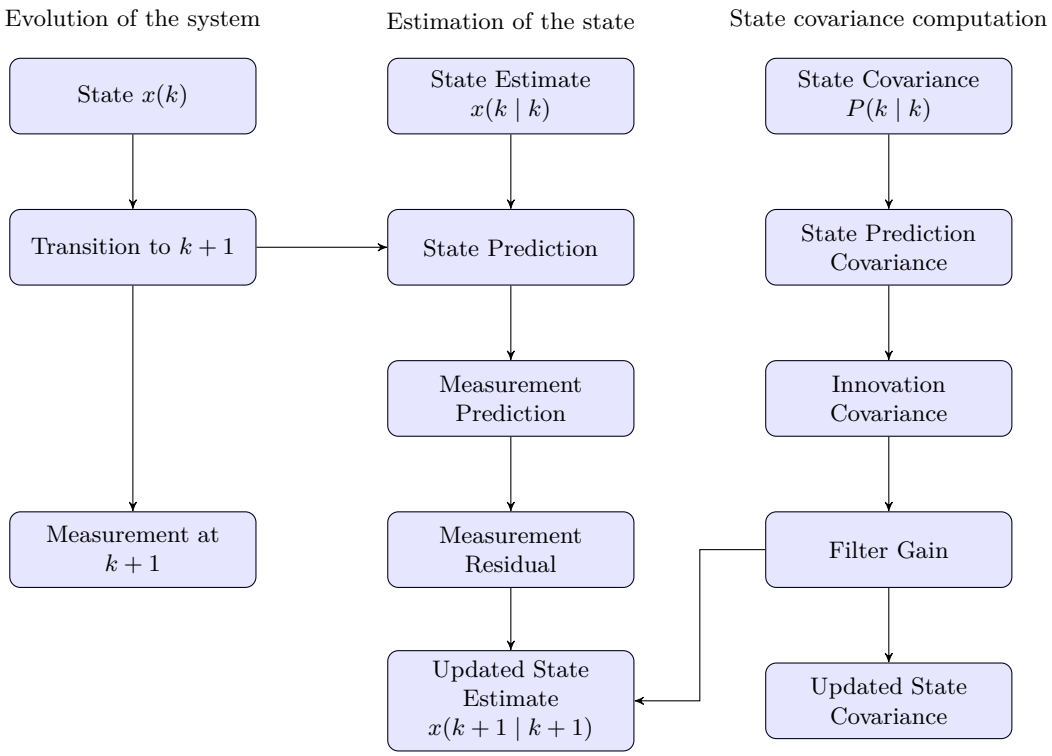


Figure B-1. Recursive estimation process using the Kalman filter.

B.2 UNSCENTED KALMAN FILTER

The unscented Kalman filter is an extension to the Kalman filter that uses the unscented transform to allow for nonlinear dynamic systems and measurement models [53]. The unscented transform approximates the distribution of a random variable y that results from an arbitrary nonlinear transformation $y = f(x)$, where the mean $\bar{x}$ and covariance P_x of $x \in \mathbb{R}^{n_x}$ are known. It does this by deterministically choosing sample points, also referred to as sigma points, defined by

$$\begin{aligned}
\chi_i &= \bar{x} & W_i &= \tfrac{\kappa}{n_x+\kappa} & i &= 0, \\
\chi_i &= \bar{x} + \left(\sqrt{(n_x+\kappa)P_x}\right)_i & W_i &= \tfrac{1}{2(n_x+\kappa)} & i &= 1,\ldots,n_x, \\
\chi_i &= \bar{x} - \left(\sqrt{(n_x+\kappa)P_x}\right)_{i-n_x} & W_i &= \tfrac{1}{2(n_x+\kappa)} & i &= n_x+1,\ldots,2n_x,
\end{aligned} \tag{B-13}$$

where κ is a scaling parameter and $\left(\sqrt{(n_x+\kappa)P_x}\right)_i$ is the ith row of the matrix square root of $(n_x+\kappa)P_x$. These sigma points capture the true mean and covariance of x. The first sigma point is called the central sigma point. They are then propagated through the function f to obtain the points $\beta_i = f(\chi_i)$, $i = 0,\ldots,2n_x$. The first two moments of y are recovered as

$$\bar{y} = \sum_{i=0}^{2n_x} W_i \beta_i, \tag{B-14}$$

$$P_y = \sum_{i=0}^{2n_x} W_i (\beta_i - \bar{y})(\beta_i - \bar{y})^T. \tag{B-15}$$

Figure B-2 illustrates how the unscented transform works for a two-dimensional problem. The original distribution is captured by five sigma points, which are then propagated through the nonlinear function, after which the mean and covariance of the new distribution are calculated empirically using Eq. (B-14) and Eq. (B-15).

In the unscented Kalman filter, both the system dynamics function

$$x(k+1) = f(x(k), u(k), v(k)) \tag{B-16}$$

and the measurement function

$$z(k+1) = h(x(k), w(k)) \tag{B-17}$$

can be nonlinear functions. Because the dynamics are nonlinear, the predicted state mean and covariance can no longer be computed using Eq. (B-6) and Eq. (B-7). Instead, $2N+1$ sigma points from the joint distribution of the state $x(k)$ and the noise $v(k)$ are drawn. Let the state portion of the sigma points be denoted χ and the noise portion of the sigma points ξ. Propagating the sigma points through f to obtain $\{\beta_i = f(\chi_i, u(k), \xi_i)\}$, the predicted state and covariance become

$$\hat{x}(k+1 \mid k) = \sum_{i=0}^{2N} W_i \beta_i, \tag{B-18}$$

$$P(k+1 \mid k) = \sum_{i=0}^{2N} W_i \left(\beta_i - \hat{x}(k+1 \mid k)\right)\left(\beta_i - \hat{x}(k+1 \mid k)\right)^T. \tag{B-19}$$

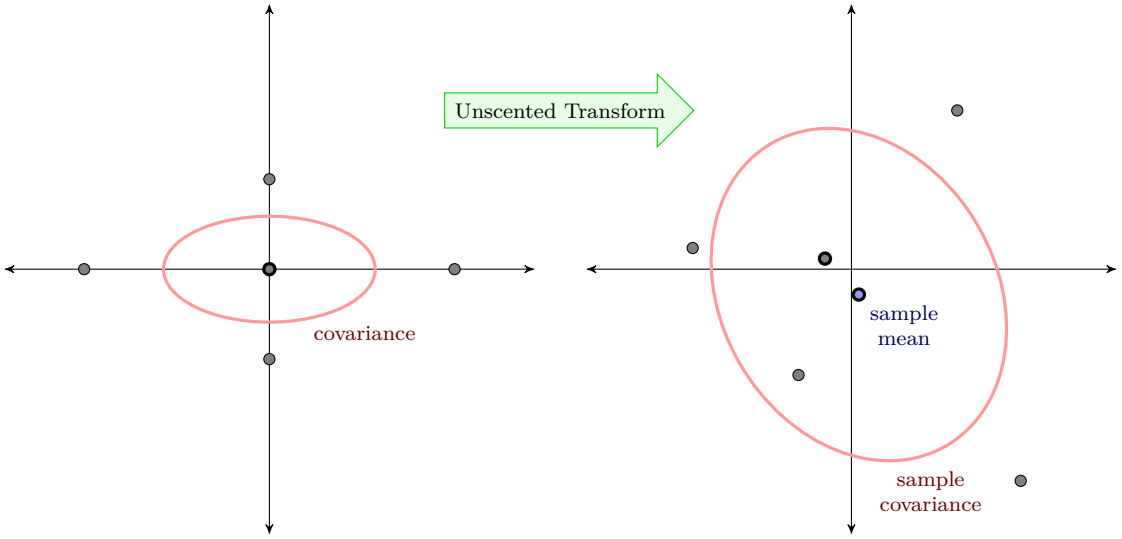


Figure B-2. The unscented transform. Sigma points are shown in gray. The central sigma point has a thicker outline.

Similarly, by sigma-point sampling from the joint distribution of $x(k)$ and $w(k)$ and propagating through the measurement function to obtain the set $\{y_i\}$, the predicted measurement and covariance are

$$\hat{z}(k+1 \mid k) = \sum_{i=0}^{2N} W_i y_i, \tag{B-20}$$

$$S(k+1) = \sum_{i=0}^{2N} W_i \left(y_i - \hat{z}(k+1 \mid k)\right) \left(y_i - \hat{z}(k+1 \mid k)\right)^T. \tag{B-21}$$

The cross-covariance needed to arrive at the filter gain and, hence, the updated state estimate and covariance is calculated in a similar way.

There are other ways of extending the Kalman filter to accomodate nonlinear transformations. The unscented Kalman filter provides one way in which probability distributions resulting from nonlinear transformations are approximated. The extended Kalman filter provides another way by not approximating the resulting probability distributions but the nonlinear transformations themselves through some kind of local approximation. This requires computing derivatives. The accuracy of the approximation is dictated by the highest order of the derivatives computed. In many applications, the unscented Kalman filter is preferable because it does not require these computations and often produces more accurate results.

B.3 HORIZONTAL TRACKER

The relative horizontal motion between the own aircraft and the intruder is estimated using an unscented Kalman filter. The tracker receives raw measurements of the form $z_{\text{raw}} = [r_{\text{slant}}\ \chi\ h_0\ h_1\ \psi_0]$,

where r_{slant} is the slant range to the intruder, χ is the relative bearing of the intruder, h_0 is the own aircraft altitude, h_1 is the intruder altitude, and ψ_0 is the own aircraft heading. All of these measurements, with the exception of the heading, are available to TCAS currently. The heading should be attainable through the onboard avionics. These raw measurements are converted to position measurements in a relative Cartesian coordinate system, $z = [x_{\text{rel}}\ y_{\text{rel}}\ h_{\text{rel}}]$, according to the follow equations:

$$\begin{cases} x_{\text{rel}} = \sqrt{r_{\text{slant}}^2 - (h_1 - h_0)^2}\sin(\theta), \\ y_{\text{rel}} = \sqrt{r_{\text{slant}}^2 - (h_1 - h_0)^2}\cos(\theta), \\ h_{\text{rel}} = h_1 - h_0, \end{cases} \tag{B-22}$$

where $\theta \equiv \chi + \psi$. The state is the position and velocities in the relative Cartesian coordinate system: $x = [x_{\text{rel}}\ y_{\text{rel}}\ h_{\text{rel}}\ \dot{x}_{\text{rel}}\ \dot{y}_{\text{rel}}\ \dot{h}_{\text{rel}}]$. The measurement function is given by the following nonlinear function:

$$z = h(x, w) = h_2(h_1(x) + w), \tag{B-23}$$

where the transformation h_1 is given by

$$[r_{\text{slant}}\ \theta\ h_{\text{rel}}] = h_1([x_{\text{rel}}\ y_{\text{rel}}\ h_{\text{rel}}\ \dot{x}_{\text{rel}}\ \dot{y}_{\text{rel}}\ \dot{h}_{\text{rel}}]) = \begin{bmatrix} \sqrt{x_{\text{rel}}^2 + y_{\text{rel}}^2 + h_{\text{rel}}^2} \\ \text{atan2}(x_{\text{rel}}, y_{\text{rel}}) \\ h_{\text{rel}} \end{bmatrix} \tag{B-24}$$

and the transformation h_2 by

$$[x_{\text{rel}}\ y_{\text{rel}}\ h_{\text{rel}}] = h_2([r_{\text{slant}}\ \theta\ h_{\text{rel}}]) = \begin{bmatrix} \sqrt{r_{\text{slant}}^2 - h_{\text{rel}}^2}\sin(\theta) \\ \sqrt{r_{\text{slant}}^2 - h_{\text{rel}}^2}\cos(\theta) \\ h_{\text{rel}} \end{bmatrix}. \tag{B-25}$$

The vector w is the zero-mean white Gaussian measurement noise vector with covariance

$$R = \begin{bmatrix} \sigma_{r_{\text{slant}}}^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_{h_{\text{rel}}}^2 \end{bmatrix}. \tag{B-26}$$

The variance σ_θ^2 is equal to $\sigma_\chi^2 + \sigma_\psi^2$, assuming the relative bearing and own aircraft heading are independent. Similarly, $\sigma_{h_{\text{rel}}}^2 = \sigma_{h_0}^2 + \sigma_{h_1}^2$. The variance in the intruder altitude measurement, $\sigma_{h_1}^2$, is $\sigma_{h_{\text{jitter}}}^2 + q^2/12$, where $\sigma_{h_{\text{jitter}}}^2$ is the contribution due to random noise and $q^2/12$ is the contribution due to quantization error, q being the amount of quantization (e.g., 25 ft or 100 ft).

The state-transition function is the following linear function:

$$\begin{aligned} x(k+1) &= Fx(k) + Bv(k) \\ &= \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 & 0 & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & 0 & \frac{1}{2}(\Delta t)^2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix} v(k), \end{aligned} \tag{B-27}$$

TABLE B-1

Parameters for the horizontal unscented Kalman filter

Parameter	Description	Typical values
$\sigma_{r_{\text{slant}}}$	std dev in the slant range	50 ft
σ_θ	std dev in the relative bearing and own aircraft heading	10 deg
$\sigma_{h_{\text{jitter}}}$	std dev in intruder altitude due to random noise	0
q	quantization in intruder altitude	25 ft
σ_{h_0}	std dev in own altitude	0
$\sigma_{\ddot{x}}$	std dev in the x position process	8 ft/s^2 (white noise) / 30 ft/s^2 (correlated)
$\sigma_{\ddot{y}}$	std dev in the y position process	8 ft/s^2 (white noise) / 30 ft/s^2 (correlated)
$\sigma_{\ddot{h}}$	std dev in the h position process	3 ft/s^2
σ_{v_0}	initial velocity std dev scale factor	250 ft/s

where $v(k)$ is the zero-mean white Gaussian process noise vector with covariance

$$Q = \begin{bmatrix} \sigma^2_{\ddot{x}_{\text{rel}}} & 0 & 0 \\ 0 & \sigma^2_{\ddot{y}_{\text{rel}}} & 0 \\ 0 & 0 & \sigma^2_{\ddot{h}_{\text{rel}}} \end{bmatrix}, \tag{B-28}$$

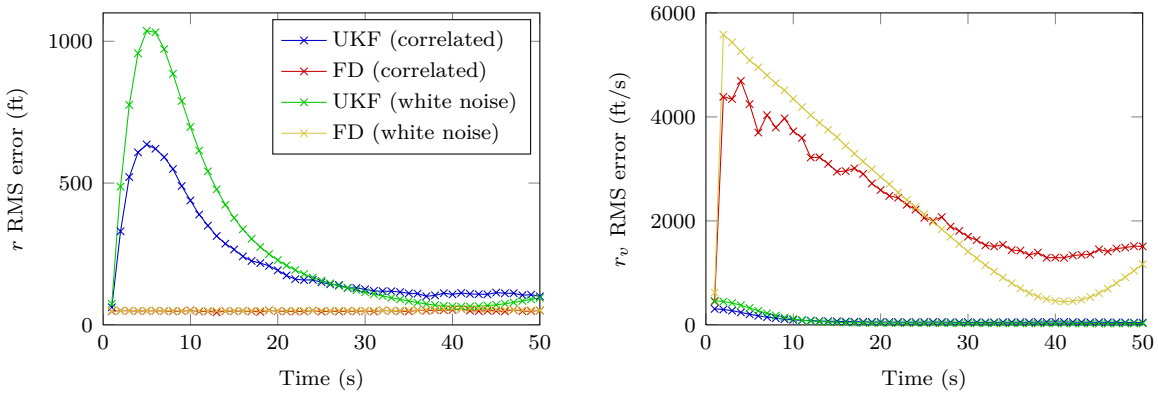
where $\ddot{x}_{\text{rel}}$, $\ddot{y}_{\text{rel}}$, and $\ddot{h}_{\text{rel}}$ represent the accelerations in the relative x, y, and h dimensions, respectively. The variances of $\ddot{x}_{\text{rel}}$, $\ddot{y}_{\text{rel}}$, and $\ddot{h}_{\text{rel}}$ are twice those of the individual aircraft, $\ddot{x}$, $\ddot{y}$, and $\ddot{h}$, respectively.

The filter is initialized using the first measurement. The mean of the velocity components of the state is initialized to zero with an arbitrary covariance of the form $\sigma^2_{v_0}\mathbb{I}_{3\times 3}$. The various parameters used in the horizontal tracker are listed in Table B-1. It also lists some typical values used when tracking white-noise and correlated model encounters.

The posterior state distribution at time k, $\mathcal{N}(x(k); \hat{x}(k \mid k), P(k \mid k))$, is a continuous distribution. Because the logic accepts discrete distributions, the posterior state distribution must be sampled to produce a collection of samples with associated probabilities. Thirteen sigma-point samples were drawn from the posterior, and each sample was converted into the uncontrolled state representation of Section 5.3 as follows:

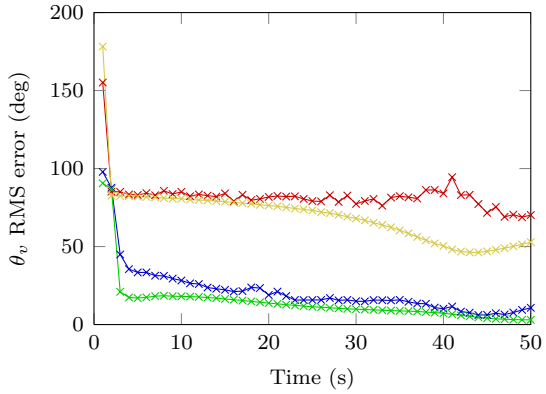
$$\begin{cases} r = \sqrt{x^2_{\text{rel}} + y^2_{\text{rel}}}, \\ r_v = \sqrt{\dot{x}^2_{\text{rel}} + \dot{y}^2_{\text{rel}}}, \\ \theta_v = \operatorname{atan2}(\dot{y}_{\text{rel}}, \dot{x}_{\text{rel}}) - \operatorname{atan2}(y_{\text{rel}}, x_{\text{rel}}). \end{cases} \tag{B-29}$$

Figure B-3 shows the root-mean-square error in r, r_v, and θ_v over the course of 50 seconds, averaged over 100,000 encounters from both the white-noise encounter model and the correlated encounter model. The white-noise encounters were produced using an 8 ft/s^2 horizontal acceleration, 3 ft/s^2 vertical acceleration. The own aircraft was equipped with the TCAS-like sensor of Section 7.1. The performance of the unscented Kalman filter is compared to a simple finite-difference



(a) r RMS error.

(b) r_v RMS error.



(c) θ_v RMS error.

Figure B-3. Horizontal tracking performance.

approach. The finite-difference approach produces better estimates of the horizontal range because it can be uniquely determined from the slant range, intruder altitude, and own altitude measurements, which are not very noisy. However, when making use of the intruder bearing measurement, which can be very inaccurate, the intruder can be mislocated, leading to large range errors. Given sufficient time, however, the horizontal range error becomes small.

B.4 VERTICAL TRACKER

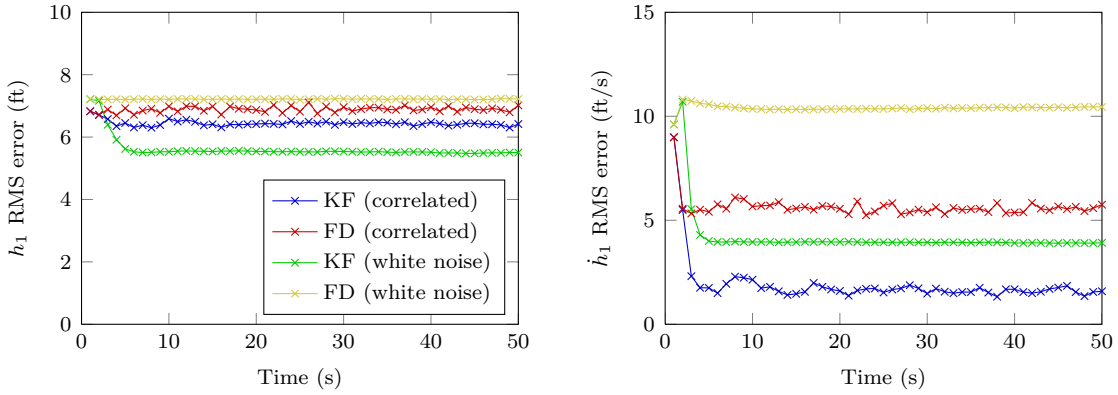
The altitude and vertical rate of the intruder are estimated using a regular Kalman filter. The tracker receives quantized measurements of the intruder altitude, $z = h_1$, from which it estimates the altitude and vertical rate, $x = [h_1 \ \dot{h}_1]$. The measurement function is

$$z = Hx + w = [1 \ 0]\, x + w. \tag{B-30}$$

TABLE B-2

Parameters for the vertical Kalman filter

Parameter	Description	Typical values
$\sigma_{h_{\text{jitter}}}$	std dev in intruder altitude due to random noise	0
q	quantization in intruder altitude	25 ft
$\sigma_{\dot{h}_1}$	std dev in intruder altitude process	3 ft/s^2
$\sigma_{\dot{h}_1}$	std dev in initial vertical rate	100 ft/s



(a) h_1 RMS error.

(b) $\dot{h}_1$ RMS error.

Figure B-4. Vertical tracking performance.

The variance of w is $\sigma_w^2 = \sigma_{h_{\text{jitter}}}^2 + q^2/12$, as in the horizontal tracker. The state-transition function, similar to Eq. (B-27), is given by

$$\begin{aligned} x(k+1) &= Fx(k) + Gv(k) \\ &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \\ \Delta t \end{bmatrix} v(k), \end{aligned} \tag{B-31}$$

where $\sigma_{v(k)}^2 = \sigma_{\ddot{h}_1}^2$.

The filter is initialized using the first measurement. The mean of the vertical rate is initialized to zero with an arbitrary variance $\sigma_{\dot{h}_1}^2$. The various parameters used in the vertical tracker are listed in Table B-2. It also lists some typical values.

Figure B-4 shows the root-mean-square error in h_1 and $\dot{h}_1$ over the course of 50 seconds, averaged over 100,000 encounters from both the white-noise encounter model and the correlated encounter model. Both aircraft in the encounters were not equipped with collision avoidance systems; therefore their vertical motion did not change in response to resolution advisories. The performance of finite-difference tracking is showed as a baseline.

B.5 DISCUSSION

This appendix presented details of the horizontal and vertical trackers used in Section 7. Although the horizontal and vertical estimation was decoupled, in reality there may be correlation between the horizontal and vertical variables that can only be captured by performing estimation of the entire state given all the measurements. This is unlikely to have a significant impact on the results. The trackers can be enhanced in various ways. For example, the vertical tracker can explicitly model the altitude quantization, instead of simply increasing the measurement noise [54]. Additionally, the distribution of the joint advisory state variable s_{RA} can serve as an indication of the "mode" of flight of the intruder. Using this extra information can result in improved estimation of the intruder altitude and vertical rate when the intruder is equipped with a collision avoidance system.

REFERENCES

[1] R. Chamlou, "Future airborne collision avoidance: Design principles, analysis plan and algorithm development," in *Digital Avionics Systems Conference*, Orlando, FL (2009), pp. 6.E.2.1–6.E.2.17.

[2] R. Chamlou, "Design principles and algorithm development for two types of NextGen airborne conflict detection and collision avoidance," in *Integrated Communications Navigation and Surveillance Conference*, Herndon, VA (2010), pp. N7.1–N7.12.

[3] G. Dowek, A. Geser, and C. Muñoz, "Tactical conflict detection and resolution in a 3-D airspace," in *4th USA/Europe Air Traffic Management R&D Seminar*, Santa Fe, New Mexico (2001).

[4] K.D. Bilimoria, "A geometric optimization approach to aircraft conflict resolution," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Denver, Colo. (2000).

[5] J.M. Shewchun, J.H. Oh, and E. Feron, "Linear matrix inequalities for free flight conflict problems," in *IEEE Conference on Decision and Control*, San Diego, CA (1997), vol. 3, pp. 2417–2422.

[6] R. Lachner, "Collision avoidance as a differential game: Real-time approximation of optimal strategies using higher derivatives of the value function," in *International Conference on Systems, Man, and Cybernetics*, Orlando, FL (1997), vol. 3, pp. 2308–2313.

[7] R. Teo and C. Tomlin, "Computing provably safe aircraft to aircraft spacing for closely spaced parallel approaches," in *Digital Avionics Systems Conference*, Philadelphia, PA (2000), vol. 1, pp. 2D2/1–2D2/9.

[8] S. Temizer, M.J. Kochenderfer, L.P. Kaelbling, T. Lozano-Pérez, and J.K. Kuchar, "Collision avoidance for unmanned aircraft using Markov decision processes," in *AIAA Guidance, Navigation, and Control Conference*, Toronto, Canada (2010).

[9] T.B. Wolf, *Aircraft collision avoidance using Monte Carlo Real-Time Belief Space Search*, Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology (2009).

[10] L.F. Winder and J.K. Kuchar, "Hazard avoidance alerting with Markov decision processes," International Center for Air Transportation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Technical Rep. ICAT-2004-4 (2004).

[11] J.K. Kuchar and L.C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems* 1(4), 179–189 (2000).

[12] M.J. Kochenderfer, J.P. Chryssanthacopoulos, L.P. Kaelbling, and T. Lozano-Perez, "Model-based optimization of airborne collision avoidance logic," Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-360 (2010), URL

http://www.ll.mit.edu/mission/aviation/publications/publication-files/atc-reports/Kochenderfer_2010_ATC-360_WW-18658.pdf.

[13] J.P. Chryssanthacopoulos, M.J. Kochenderfer, and R.E. Williams, "Improved Monte Carlo sampling for conflict probability estimation," in *AIAA Non-Deterministic Approaches Conference*, Orlando, Florida (2010).

[14] L.C. Yang and J.K. Kuchar, "Prototype conflict alerting system for free flight," *Journal of Guidance, Control, and Dynamics* 20(4), 768–773 (1997).

[15] B.D. Carpenter and J.K. Kuchar, "Probability-based collision alerting logic for closely-spaced parallel approach," in *AIAA Aerospace Sciences Meeting*, Reno, NV (1997).

[16] R. Bellman, *Dynamic Programming*, Princeton: Princeton University Press (1957).

[17] M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley series in probability and mathematical statistics, New York: Wiley (1994).

[18] D.P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, Belmont, Mass.: Athena Scientific, 3rd ed. (2005).

[19] O. Sigaud and O. Buffet (eds.), *Markov Decision Processes in Artificial Intelligence*, Hoboken, NJ: Wiley (2010).

[20] W.B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Hoboken, NJ: Wiley (2007).

[21] D.P. Bertsekas and J.N. Tsitsiklis, *Neuro-Dynamic Programming*, Optimization and neural computation series, Belmont, Mass.: Athena Scientific (1996).

[22] R. Munos and A. Moore, "Variable resolution discretization in optimal control," *Machine Learning* 49(2–3), 291–323 (2002).

[23] S. Davies, "Multidimensional triangulation and interpolation for reinforcement learning," in M.C. Mozer, M.I. Jordan, and T. Petsche (eds.), *Advances in Neural Information Processing Systems*, Cambridge, Mass.: MIT Press, vol. 9, pp. 1005–1011 (1997).

[24] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence* 101(1–2), 99–134 (1998).

[25] E.W. Kamen and J.K. Su, *Introduction to Optimal Estimation*, London: Springer (1999).

[26] C.H. Papadimitriou and J.N. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of Operations Research* 12(3), 441–450 (1987).

[27] W.S. Lovejoy, "Computationally feasible bounds for partially observed Markov decision processes," *Operations Research* 39(1), 162–175 (1991).

[28] T. Smith and R.G. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Conference on Uncertainty in Artificial Intelligence*, Edinburgh, Scotland (2005), pp. 542–547.

[29] H. Kurniawati, D. Hsu, and W. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Proc. Robotics: Science and Systems* (2008).

[30] M.L. Littman, A.R. Cassandra, and L.P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *International Conference on Machine Learning*, Tahoe City, CA (1995), pp. 362–370.

[31] M. Hauskrecht, "Value-function approximations for partially observable Markov decision processes," *Journal of Artificial Intelligence Research* 13, 33–94 (2000).

[32] J.L. Fernández, R. Sanz, R.G. Simmons, and A.R. Diéguez, "Heuristic anytime approaches to stochastic decision processes," *Journal of Heuristics* 12(3), 181–209 (2006).

[33] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online planning algorithms for POMDPs," *Journal of Artificial Intelligence Research* 32, 663–704 (2008).

[34] D. Nikovski and I. Nourbakhsh, "Learning probabilistic models for decision-theoretic navigation of mobile robots," in *International Conference on Machine Learning*, Stanford, CA (2000), pp. 671–678.

[35] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, Cambridge, Mass.: MIT Press (1998).

[36] International Civil Aviation Organization, "Surveillance, radar and collision avoidance," in *International Standards and Recommended Practices*, vol. IV, annex 10, 4th ed. (2007).

[37] D.W. Moore, *Simplical Mesh Generation with Applications*, Ph.D. thesis, Cornell University (1992).

[38] RTCA, "Minimum operational performance standards for Traffic Alert and Collision Avoidance System II (TCAS II)," DO-185B (2008).

[39] M.J. Kochenderfer, M.W.M. Edwards, L.P. Espindle, J.K. Kuchar, and J.D. Griffith, "Airspace encounter models for estimating collision risk," *Journal of Guidance, Control, and Dynamics* 33(2), 487–499 (2010).

[40] M.J. Kochenderfer, L.P. Espindle, J.K. Kuchar, and J.D. Griffith, "Correlated encounter model for cooperative aircraft in the national airspace system," Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-344 (2008), URL `http://www.ll.mit.edu/mission/aviation/publications/publication-files/atc-reports/Kochenderfer_2008_ATC-344_WW-18099.pdf`.

[41] R.E. Neapolitan, *Learning Bayesian Networks*, Upper Saddle River, NJ: Prentice Hall (2004).

[42] G.N. Iyengar, "Robust dynamic programming," *Mathematics of Operations Research* 30(2), 257–280 (2005).

[43] J.K. Kuchar and A.C. Drumm, "The Traffic Alert and Collision Avoidance System," *Lincoln Laboratory Journal* 16(2), 277–296 (2007).

[44] Y. Bar-Shalom, X.R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, New York: Wiley (2001).

[45] G. Minkler and J. Minkler, *Theory and Applications of Kalman Filtering*, Palm Bay, FL: Magellan Book Company (1993).

[46] S.G. Mohinder and A.P. Andrews, *Kalman Filtering: Theory and Practice Using MATLAB*, New York: Wiley (2001).

[47] C.V. Goldman and S. Zilberstein, "Optimizing information exchange in cooperative multi-agent systems," in *International Joint Conference on Autonomous Agents & Multiagent Systems*, Melbourne, Australia (2003), pp. 137–144.

[48] C.V. Goldman and S. Zilberstein, "Decentralized control of cooperative systems: Categorization and complexity analysis," *Journal of Artificial Intelligence Research* 22, 143–174 (2004).

[49] F.A. Oliehoek, *Value-based planning for teams of agents in stochastic partially observable environments:*, Ph.D. thesis, University of Amsterdam (2010).

[50] S.J. Russell and A. Zimdars, "Q-decomposition for reinforcement learning agents," in T. Fawcett and N. Mishra (eds.), *International Conference on Machine Learning*, Washington DC (2003), pp. 656–663.

[51] J.K. Rosenblatt, "Optimal selection of uncertain actions by maximizing expected utility," *Autonomous Robots* 9(1), 17–25 (2000).

[52] T.B. Billingsley, L.P. Espindle, and J.D. Griffith, "TCAS multiple threat encounter analysis," Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-359 (2009).

[53] S.J. Julier and J.K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE* 92(3), 401–422 (2004).

[54] J.W. Andrews, "An improved technique for altitude tracking of aircraft," Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-105 (1981).