

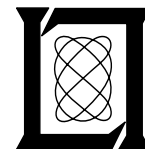
**Project Report
ATC-356**

**Unmanned Aircraft Collision Avoidance Using
Partially Observable Markov Decision Processes**

S. Temizer
M.J. Kochenderfer
L.P. Kaelbling
T. Lozano-Perez
J.K. Kuchar

22 September 2009

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS



This research effort was supported under the auspices of the Lincoln Laboratory Advanced Concepts Program. The Advanced Concepts Program is supported principally by the Department of the Air Force under Contract FA8721-05-C-0002.

Approved for public release; distribution is unlimited.


This research effort was supported under the auspices of the Lincoln Laboratory Advanced Concepts Program. The Advanced Concepts Program is supported principally by the Department of the Air Force under Contract FA8721-05-C-0002.

This report may be reproduced to satisfy needs of U.S. Government agencies.

The 66ABW Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER


Gary Tutungian
Administrative Contracting Officer
Plans and Programs Directorate
Contracted Support Management

Non-Lincoln Recipients

PLEASE DO NOT RETURN

Permission has been given to destroy this document when it is no longer needed.

Massachusetts Institute of Technology
Lincoln Laboratory

Unmanned Aircraft Collision Avoidance Using Partially Observable Markov Decision Processes

M.J. Kochenderfer
Group 42

J.K. Kuchar
Group 43

S. Temizer
L.P. Kaelbling
T. Lozano-Perez
MIT CSAIL

Project Report ATC-356

22 September 2009

Approved for public release; distribution is unlimited.

Lexington

Massachusetts

This page intentionally left blank.

ABSTRACT

Before unmanned aircraft can fly safely in civil airspace, robust airborne collision avoidance systems must be developed. Instead of hand-crafting a collision avoidance algorithm for every combination of sensor and aircraft configuration, this project investigates the automatic generation of collision avoidance logic given models of aircraft dynamics, sensor performance, and intruder behavior. By formulating the problem of collision avoidance as a partially-observable Markov decision process (POMDP), a generic POMDP solver can be used to generate avoidance strategies that optimize a cost function that balances flight-plan deviation with collision. Experimental results demonstrate the suitability of such an approach using three different sensor modalities and two aircraft performance models.

This page intentionally left blank.

ACKNOWLEDGMENTS

The Lincoln Laboratory portion of this work was supported under Air Force contract number FA8721-05-C-0002. The MIT CSAIL portion of this work was supported by the Office of the Director of Defense Research and Engineering. Interpretations, opinions, and conclusions are those of the authors and do not reflect the official position of the United States government. The authors wish to thank Nikhil Khanna for his contributions to the early stages of this work while he was an undergraduate at MIT. Special thanks are due to Dan Griffith, an associate staff member at Lincoln Laboratory, who has been tremendously helpful in integrating the POMDP collision avoidance system into the Collision Avoidance System Safety Assessment Tool (CASSATT).

This page intentionally left blank.

TABLE OF CONTENTS

	Page
Abstract	iii
Acknowledgments	v
List of Illustrations	ix
List of Tables	xi
 1. INTRODUCTION	 1
 2. POMDPS	 3
2.1 Formulation	3
2.2 Solution Methods	4
 3. MODEL	 7
3.1 State, Action, and Observation Spaces	7
3.2 Reward Function	8
3.3 Observation Model	9
3.4 State-Transition Model	12
 4. SIMULATION	 15
4.1 Encounter Model	16
4.2 Dynamic Model	16
4.3 Sensor Model	16
4.4 State Estimation	17
4.5 Policy Evaluation	20
 5. RESULTS	 25
 6. DISCUSSION	 29
6.1 Discretization	29

6.2	Parameter Values	29
6.3	Missing State Information	29
6.4	Observation Models	30
6.5	Estimation of Intruder Vertical Velocity	30
6.6	Estimation of Closure Rate	31
6.7	Three-Dimensional Trajectory Extrapolation	31
7.	CONCLUSIONS	33
8.	FUTURE WORK	35
A.	PSEUDOCODE	37
B.	POMDP GENERATION	43
C.	ENCOUNTER ANALYZER	45

LIST OF ILLUSTRATIONS

Figure No.		Page
1	Qualitative performance characteristics of various sensor modalities.	1
2	Simulation framework.	15
3	Sensor capabilities.	17
4	A screenshot of the Encounter Analyzer using the TCAS sensor.	27
5	A screenshot of the Encounter Analyzer using the radar sensor.	27
6	A screenshot of the Encounter Analyzer using the EO/IR sensor.	28
7	Gaussian distribution approximated by four flat distributions stacked on top of each other.	30
8	Path of intruder and discretized observations.	31
C-1	A screenshot of the Encounter Analyzer showing flight trajectory and debugging information.	46

This page intentionally left blank.

LIST OF TABLES

Table No.		Page
1	Sensor error model characteristics.	11
2	Intruder aircraft horizontal acceleration model.	13
3	Intruder aircraft vertical acceleration models.	14
4	Specification parameters.	18
5	POMDP processing for the TCAS sensor.	20
6	Histogram showing the frequency of number of end states for the TCAS sensor.	21
7	POMDP processing for the perfect sensor.	22
8	Histogram showing the frequency of number of end states for the perfect sensor.	22
9	Risk ratio for different sensors and aircraft performance constraints.	26
10	Risk ratios for analytic collision avoidance system.	32

This page intentionally left blank.

1. INTRODUCTION

Because of the potential for commercial, military, law-enforcement, and scientific applications, unmanned aircraft have been receiving considerable attention in recent years. However, unmanned aircraft are not currently permitted access to civil airspace in the United States without special permission from the Federal Aviation Administration (FAA). One of the primary concerns with integrating unmanned aircraft is their inability to robustly sense and avoid other aircraft. Although sensor information can be transmitted to a ground pilot who can then maneuver the aircraft to avoid collision, there are concerns about communication latency and reliability. In order to provide the high level of safety required by the FAA, an automated airborne collision avoidance system is likely to be necessary.

The deployment of any collision avoidance system requires a lengthy development process followed by a rigorous certification process. Development of the Traffic-alert and Collision Avoidance System (TCAS), currently mandated onboard all large transport aircraft worldwide, started in the 1950s but was not certified for operational use until relatively recently. The system issues vertical rate resolution advisories to pilots who are then responsible for maneuvering the aircraft. TCAS is not certified for autonomous use, and it is likely that the certification of an autonomous system will require even more extensive testing and analysis.

Further complicating the certification process of collision avoidance systems for unmanned aircraft is the diversity of their aircraft performance characteristics and sensor capabilities. Unmanned aircraft can range from under a pound to many tons with wildly varying flight dynamics. Several sensor modalities have been considered for supporting collision avoidance, including electro-optical/infrared (EO/IR), radar, TCAS, and Automatic Dependent Surveillance-Broadcast (ADS-B). As Figure 1 illustrates, these sensor modalities vary in their capabilities. It would be very difficult to develop and certify a different collision avoidance system for every combination of sensor configuration and aircraft platform. Current efforts in the unmanned aircraft industry have focused on proprietary solutions for specific platforms and sensors, but a common system that would accommodate different sensor configurations and flight characteristics would significantly reduce the cost of development and certification.

Modality	Measurement Accuracy			Coverage		
	Range	Azimuth	Elevation	FOV	Range	Traffic
TCAS	■	■	■	■	■	■
Radar	■	■	■	■	■	■
EO/IR	■	■	■	■	■	■
ADS-B	■	■	■	■	■	■

Figure 1. Qualitative performance characteristics of various sensor modalities. Green, yellow, and red indicate good, moderate, and poor performance, respectively.

Such a system would take as input models of the flight dynamics, intruder behavior, and sensor characteristics and attempt to optimize the avoidance strategy so that a predefined cost function is minimized. The cost function could take into account competing objectives, such as flight plan adherence and avoiding collision. One way to formulate a problem involving the optimal control of a stochastic system with uncertain observations is as a partially-observable Markov decision process (POMDP). POMDPs have been studied in the operations research and artificial intelligence communities, but only in the past few years have generic POMDP solution methods been developed that can find approximate solutions to problems with moderate to large state spaces in reasonable time. The aim of this project is to investigate the feasibility of applying state-of-the-art POMDP solution methods to the problem of collision avoidance.

The remainder of this report is organized as follows. Section 2 reviews the POMDP formulation and solution methods. Section 3 describes how the problem of collision avoidance may be represented using a reward function, observation model, and state-transition model. Section 4 describes the simulation framework used to evaluate the POMDP policies. Section 5 presents the results from a collection of experiments involving different flight dynamics and sensor capabilities. Section 6 elaborates on limitations of developed models and suggests ways to improve them. Section 7 discusses some of the main conclusions from this work and lessons learned. Section 8 describes future work. Appendix A contains the pseudocode referred to in the main text. Appendix B describes the POMDP Generator, a software package that produces a discrete POMDP representation given a set of parameters describing aircraft behavior and the capabilities of the sensors. Appendix C describes the Encounter Analyzer, a software package for visualizing and debugging encounter scenarios.

2. POMDPS

A Markov decision process (MDP) is a stochastic process where the state of the system changes probabilistically according to the current state and action. POMDPs extend MDPs by including an observation process that probabilistically generates observations conditioned on the current state and action. The solution to a POMDP is a *policy*, or way of behaving, that selects actions in a way that takes into account both the current uncertainty about the underlying state of the system (e.g., exact relative position of the intruder aircraft), as well as future uncertainty about how the system state will evolve (e.g., what kinds of maneuvers the other aircraft will make). The objective is to maximize the expected accumulation of reward (or minimizes the expected accumulation of cost).

Several formulations of POMDPs have been studied in the literature, but this project focuses on the discrete-time formulation with discrete state and action spaces. This section briefly presents a POMDP formulation and discusses solution techniques.

2.1 FORMULATION

This report uses \mathcal{S} to represent the state space, \mathcal{A} to represent the action space, and Ω to represent the observation space, all assumed discrete. The *state-transition function* $T : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$ determines the distribution over the next state given the current state and action taken. The probability of transitioning to state s' after taking action a from state s is written $T(s, a, s')$. The *observation function* $O : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\Omega)$ determines the distribution over the observations received after taking some action resulting in state s' . The probability of receiving observation o after taking action a landing in state s' is written $O(s', a, o)$.

In general, the initial state is unknown. The uncertainty in the initial state is represented by a belief state, which is a probability distribution over states $b : \mathcal{S} \rightarrow \mathbb{R}$. The probability of being in state s is written $b(s)$. The space of possible belief states is denoted \mathcal{B} . The initial belief state is denoted b_0 and is updated with each observation according to Bayes' rule. If the current belief state is b and action a is taken resulting in an observation o , the new belief state b' is given by

$$\begin{aligned} b'(s) &= \Pr(s' \mid o, a, b) \\ &\propto \Pr(o \mid s', a, b) \Pr(s' \mid a, b) \\ &= \Pr(o \mid s', a) \sum_{s' \in \mathcal{S}} T(s, a, s') b(s) \\ &= O(s', a, o) \sum_{s' \in \mathcal{S}} T(s, a, s') b(s). \end{aligned}$$

The belief-update process is often referred to as *state estimation*.

Given the current belief state, the objective is to choose an action that maximizes the *expected discounted return*. The discounted return for a sequence of states s_t and actions a_t is given by

$$\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t), \quad (1)$$

where $\gamma \in [0, 1)$ is a *discount factor* and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the *reward function*. The reward for taking action a from state s is written $R(s, a)$.

The solution to a POMDP is a *policy* $\pi : \mathcal{B} \rightarrow \mathcal{A}$ that specifies which action maximizes the expected discounted reward given a belief state. It is known that optimal policies can be represented as a collection of α -vectors, denoted Γ . Each α -vector is a vector consisting of $|\mathcal{S}|$ components and is associated with a particular action. The expected discounted return when starting with belief b is

$$V(b) = \max_{\alpha \in \Gamma} (\alpha \cdot b), \quad (2)$$

where $\alpha \cdot b$ is the inner product of an α -vector with a vector representation of the belief state. The function V is known as the *value function*. The policy evaluated at belief state b is the action associated with the α -vector that maximizes the inner product.

2.2 SOLUTION METHODS

Finding the collection of α -vectors that represents the optimal policy can be challenging, even for relatively small problems. A variety of exact solution methods can be found in the literature, but generally these methods do not scale well to large problems. Approximate solution methods generally scale much better and many of them provide bounds on the *regret* for the policies they find. The regret of a policy π is the difference between the expected discounted return starting at b_0 when following π and the expected discounted return starting at b_0 when following an optimal policy π^* .

In recent years, point-based methods for finding approximate solutions to POMDPs have received attention because of their ability to solve problems that are orders of magnitude larger than was previously possible. Point-based methods involve sampling from the belief space \mathcal{B} . The more successful point-based methods focus the sampling of belief states. Currently, it appears that the algorithm that offers the best overall performance is SARSOP, which stands for Successive Approximations of the Reachable Space under Optimal Policies [1]. This project originally used HSVI2, which stands for Heuristic Search Value Iteration [2], but SARSOP performed better on our problems.

An implementation of SARSOP is publicly available and we were able to use the software without any modification. SARSOP takes as input a standard textual representation of a POMDP, including γ , b_0 , R , T , and O . When the regret bounds fall below some preset value or a user interrupts the solution process, SARSOP outputs a policy file represented as a collection of α -vectors.

Crucially, although it may require considerable computation to find a near-optimal policy, this work is done offline. Once a policy has been computed, it can be executed very efficiently

online. In the course of this project we have developed a new algorithmic technique to make the execution process even more efficient, making it entirely suitable for execution online, in real time, on an aircraft.

Although this project has focused entirely on finding α -vectors offline, there are other approaches to finding and representing policies. Online approaches decide what action to execute by searching only from the current belief state, instead of trying to find a comprehensive policy that is optimal for all belief states [3]. From the current belief state, these methods explore different action sequences up to some horizon and then select the sequence that results in the largest expected discounted return. Computing the expected discounted return for an action sequence involves updating the belief state based on hypothetical measurements obtained with each state transition. The search space can quickly become intractable as the time horizon increases and the state, action, and observation sets grow. Methods exist that allow the time horizon to be kept short and sampling techniques can be used to manage large state, action, and observation spaces. One concern with an online method that involves sampling is the nondeterminism of the resulting behavior. However, a large collection of online runs can be used to train a function approximator, such as a neural network, offline. The function approximator can then be used online to recommend actions. Further research may involve investigating some of these approaches.

This page intentionally left blank.

3. MODEL

The true state space model in the collision avoidance problem is continuous and consists of the following components for both aircraft involved in the encounter:

- Position specified in a global (earth) coordinate system with east, north, and altitude axes
- Orientation specified as yaw, pitch, and roll angles
- Air speed and air speed acceleration
- Vertical rate and vertical acceleration
- Yaw rate, pitch rate, and roll rate

This set of components is referred to as the aircraft state vector. The collection of all possible states forms a 26-dimensional continuous space. Most POMDP solution methods, however, require the sets of states, actions, and observations to be finite. In order to apply these methods, the space must be discretized. The size of a discretized state space is exponential in the number of dimensions. Discretizing each of the 26 dimensions into only two bins results in an impractical number of states, which requires us to project down to a much lower dimensional space that captures the essence of the encounter. This section discusses how this is done.

3.1 STATE, ACTION, AND OBSERVATION SPACES

In this pilot study of applying POMDPs to the collision avoidance problem, we consider a simplified version of the problem in which the aircraft can only maneuver vertically. In order to maintain safety, we conservatively treat all intruders, whatever their azimuth, as if they were in the azimuthal plane defined by our current heading.

State space To represent the relative positions and velocities of the aircraft, we used a two-dimensional coordinate system, called the *relative coordinate system* (RCS) with our aircraft at the origin. The X -axis of this system is a horizontal ray, with azimuth equal to our heading. The Y -axis is altitude. In this system, the state consists of the following components:

- X : horizontal distance from own aircraft to other aircraft;
- Y : vertical distance from own aircraft to other aircraft;
- $Relative Vx$: velocity in X , representing the horizontal closure rate;
- $Other Vy$: absolute vertical velocity of other aircraft; and
- $Own Vy$: absolute vertical velocity of own aircraft.

This 5-dimensional state space is discretized by dividing each dimension into a finite number of bins. The sizes of the bins may be non-uniform. The overall state-space is then a set of 5-dimensional boxes (or hyperrectangles) that exhaust a continuous piece of the overall 5-dimensional state space. We augment the state space with two special states: a *START* state and a *DONE* state. These states help model situations when the state space is initialized (and the encounter has not started), and when the encounter is over, respectively. There are multiple *START* and *DONE* states, one for each of our (discretized) vertical velocity values. Having discretized the state space in this way, a state may be represented simply as an index into the set of boxes spanning the space.

Action space We adopted a simple discrete action-space model that consists of commands to our aircraft to apply positive or negative vertical accelerations. In the experiments described in this report we used three actions: accelerate up (at a fixed rate for a fixed duration), accelerate down (at the same fixed rate and duration), or maintain vertical velocity. It would be very straightforward to consider a larger set of acceleration command values.

Observation space The discrete model of the observation space is constructed in a way similar to the discrete state space. There are two types of observational information: our vertical velocity (which we assume is always completely and correctly observed) and possible single detection of an intruder using a sensor system. We are considering aircraft with two different kinds of sensors: the radar and TCAS sensors both report observations of intruder position in the relative coordinate system; the EO/IR sensor reports only the elevation angle of the intruder.

In our current model, observations for the radar and TCAS sensors are discretized into the same bins as the X and Y components of the state space. The model could easily be changed to provide observations at a higher or lower granularity. Observations for the EO/IR sensor are particular angle values that can be thought of as the centers of angular bins, which are again not necessarily uniform. In addition, for both sensors, there is a special *noObs* observation for the case when no intruder is detected.

3.2 REWARD FUNCTION

The reward function in our POMDP is in the form of costs (or negative rewards) rather than positive rewards. It is designed with the following three objectives in mind:

- As the primary goal of the collision avoidance algorithm, the intruder should never occupy the same bin as our aircraft in the RCS, which implies a collision or a very dangerous encounter. Note that our aircraft resides at the origin of the RCS, and it is possible that the origin might be on the edge or vertex of one or more bins rather than being inside a single bin due to the chosen vertical and horizontal division strategy. In that case, the collision avoidance algorithm should prevent the intruder from moving into any one of the bins that have any boundaries touching the origin.

- In addition to preventing collision, it is desirable to maintain some protected airspace around own aircraft where the other aircraft should not penetrate. This protected airspace is specified by two parameters: a vertical separation range and a horizontal separation range. In our tests, we used 200 ft vertical and 1000 ft horizontal separation ranges, double that of the near mid-air collision (NMAC) definition used in prior TCAS safety studies [4–7]. The second goal of the collision avoidance algorithm should be to prevent other aircraft moving into any bin that has some parts overlapping with the protected space.
- As the last goal, if there is no danger of collision or penetration of protected airspace, we should level off and try to maintain a zero vertical velocity.

In order to satisfy these goals, the reward may be specified as a function of the state of the system. It is specified using three user-defined parameters:

- *Collision cost*: The cost of any state in which the intruder is in the same X and Y bins as our aircraft, currently set to -1000 ;
- *Protected airspace violation cost*: The cost of any state in which the other aircraft is within the protected airspace region in X and Y , currently set to -500 ; and
- *Vertical velocity penalty*: The cost for being in a state where the $OwnVy$ bin does not contain 0; currently set to -0.1 .

All other states are assumed to have zero reward. Note that the solution to the POMDP will remain the same for any linear scaling of all reward values, so only the relative magnitudes have an effect.

In order to emphasize the importance of avoiding crashes at any time (rather than simply trying to postpone them), we might wish to have a discount factor of 1.0. Doing so, however, violates the mathematical assumptions of the POMDP formulation, and so we set it to 0.99, which has the desired effect.

3.3 OBSERVATION MODEL

The observation model of a POMDP specifies $\Pr(o \mid s)$, that is, the probability of making an observation o , given that the actual state is s . We assume that, on every step, the observation has two components: o_{ovy} , our measured vertical velocity, and o_d , the observed detection of the intruder, and that these are independent, so

$$\Pr(o_{ovy}, o_d \mid s) = \Pr(o_{ovy} \mid s) \Pr(o_d \mid s).$$

The measurement of our vertical velocity is always accurate, so $\Pr(o_{ovy} \mid s) = 1$ if o_{ovy} is equal to the $OwnVy$ component of s , and 0 otherwise.

The observed detection is more complex. There are three potential sources of noise:

- *false positives*: we may detect an intruder when in fact there is no intruder;
- *false negatives*: we may fail to detect an intruder when in fact one is present; and
- *measurement error*: we may detect an intruder in a position or at angle that is not correct.

We assume fixed probabilities for false positives p_{fp} and false negatives p_{fn} , and assume that if there is a false positive detection, it is generated with uniform probability over the space of values of o_d . In the models used in the experiments $p_{fp} = p_{fn} = 0.01$ for the radar and EO/IR sensors. For the TCAS sensor, $p_{fp} = 0$ and $p_{fn} = 0.01$. For the perfect sensor, $p_{fp} = p_{fn} = 0$.

When s is a START or DONE state (the encounter has not yet begun or has terminated) or when $Y > \text{maxRange}$, that is, when the distance to the other aircraft is greater than the range of the sensor, then $\Pr(o_d = \text{noObs} \mid s) = 1 - p_{fp}$. That is, with high probability, the observation is *noObs*. We used a value of 5 nautical miles for *maxRange* for all sensors. For any other observation $\Pr(o_d = d \mid s, fp) = |O_d|^{-1}$; that is, it is uniform over the space of possible actual detection observations.

Finally, if the intruder is within the modeled volume of the state space, there is some chance of not seeing the intruder: $\Pr(o_d = \text{noObs} \mid s) = p_{fn}$. Otherwise, with probability $1 - p_{fn}$, we make a detection d .

Detections from TCAS, radar, and perfect sensors all correspond to bins in the X, Y dimensions of the state space, and they are handled in the same way. In each case, a *margin* (the size of which depends on the sensor) is added to all four sides of the X, Y rectangle corresponding to the detection d . Then we consider all of the X, Y bins b_i that overlap the expanded detection bin, and the proportion of the expanded detection bin that overlaps b_i , called p_i . So,

$$\Pr(o_d = d \mid s) = (1 - p_{fp})p_i + p_{fp}|O_d|^{-1} ,$$

for any state in which the intruder is in X, Y bin b_i , for all bins b_i , and

$$\Pr(o_d = d \mid s) = p_{fp}|O_d|^{-1}$$

otherwise.

The perfect sensor has zero margin. For the TCAS sensor, the margin is defined in terms of standard TCAS sensor error parameters (shown in table 1):

$$\begin{aligned} \text{Margin} = & \text{measured altitude quantization} + \\ & 3 \cdot \text{range error standard deviation} + \\ & 3 \cdot \text{altimetry error scale} \end{aligned}$$

The 3s in the margin calculation was chosen somewhat arbitrarily and can be tuned. It should be large enough so that the margin covers the region from which a noisy sensor reading may have originated, but it should be small enough to allow the POMDP to properly localize the intruder.

TABLE 1
Sensor error model characteristics.

TCAS		
Altitude quantization	25 ft	The altitude of the other aircraft is measured with some quantization.
Range error σ	50 ft	Range error is Gaussian with zero mean.
Altimetry error λ	40 ft	Altimetry error is Laplacian with zero mean. Altimetry error bias remains constant during an encounter.
Radar		
Range error σ	50 ft	Range error is Gaussian with zero mean.
Elevation error σ	1 deg	Elevation error is Gaussian with zero mean.
EO/IR		
Elevation error σ	0.5 deg	Elevation error is Gaussian with zero mean.
Perfect		
No error.		

The margin for the radar sensor is defined in terms of standard radar error parameters (shown in table 1):

$$\text{Margin} = 3 \cdot \text{range error standard deviation} + \text{longest distance to bin} \times \tan(3 \cdot \text{elevation error standard deviation})$$

Detections from the EO/IR sensor are nominal angles. For each state s in which the intruder is located in the modeled X, Y space, we can compute a *nominal* elevation angle $d^*(s)$ to the intruder. We assume that the probability of observing a detection angle d when the actual angle is d^* is proportional to a Gaussian density with mean at d^* ; so,

$$\Pr(o_d = d \mid s) = (1 - p_{fp}) \cdot \frac{1}{z} e^{(d-d^*(s))^2} + p_{fp} |O_d|^{-1} ,$$

where

$$z = \sum_s e^{(d-d^*(s))^2}$$

is the normalization constant.

The extended observation functions for the TCAS, radar, EO/IR, and perfect sensors are listed in Appendix A as Algorithms 2, 3, 4, and 5, respectively. The pseudocode does not include various minor routines such as normalization and scaling of the list of probabilities.

3.4 STATE-TRANSITION MODEL

The initial state distribution specifies that the system starts in a *START* state. At each step, an action is taken and the probability distribution over the state space is updated according to the state transition model.

Our assumption is that there is no actual stochasticity in the dynamics of the system. However, we model the uncertainty in intruder behavior as a random process; and the fact that the state space is discretized will introduce uncertainty in the transitions, even though they are governed by a deterministic physical process.

Our state transition model is characterized by the following parameters:

- Controller frequency, ΔT : Duration between successive consultations of the POMDP policy for choosing an action. This value is used by the POMDP formulation to predict what the state will be in the next iteration.
- Magnitude of our vertical acceleration, $OwnAy$.
- Our vertical velocity limits, $OwnVyMin$ and $OwnVyMax$.
- Probability of staying in *START* state when already in *START* state.
- Probability of appearing in any other state when in *START* state.
- Intruder horizontal and vertical acceleration model (details follow).

For the intruder horizontal acceleration model, we use the distribution in Table 2. For the vertical acceleration, we investigated three different models. The first is *Random Walk*, where the other aircraft is oblivious to own aircraft, or we have no idea about the intention of the other aircraft. The second is *Risk Aversive*, where the other aircraft tries to increase vertical separation, similar to TCAS in a non-crossing Resolution Advisory. The third is *Hostile*, where other aircraft tries to decrease vertical separation. The parameters used for these three models are summarized in Table 3.

Given these parameters, we can compute $\Pr(s' | s, a)$ as follows.

- First, we consider each possible pair of vertical and horizontal accelerations a_o that might be chosen by the other aircraft, and compute their probabilities p_o as the product of the probabilities in the intruder acceleration models.
- For each vertex of the bin s , we determine how that particular point in state space would be transformed given the execution of our own acceleration a , and the intruder accelerations a_o .
- The result is a new box, B , in 5-dimensional space. For each new state s' , we can compute the percentage of B that overlaps s ; that overlap percentage is $\Pr(s' | s, a, a_o)$. Any probability mass outside the boundaries of the modeled state space is assigned $\Pr(done, OwnVy | s, a, a_o)$.

TABLE 2
Intruder aircraft horizontal acceleration model.

\dot{v} (ft/s ²)	Pr
−30.0	0.10
−20.0	0.16
−10.0	0.16
0.0	0.16
10.0	0.16
20.0	0.16
30.0	0.10

- Finally,

$$\Pr(s' \mid s, a) = \sum_{a_o} \Pr(s' \mid s, a, a_o) p_o .$$

Algorithm 6 (Appendix A) provides pseudocode for the Extended Transition Function.

TABLE 3

Intruder aircraft vertical acceleration models.

Intruder Above		Intruder Below	
\ddot{h} (ft/s ²)	Pr	\ddot{h} (ft/s ²)	Pr
Random Walk			
−10.0	0.1	−10.0	0.1
−5.0	0.2	−5.0	0.2
0.0	0.4	0.0	0.4
5.0	0.2	5.0	0.2
10.0	0.1	10.0	0.1
Risk Aversive			
−5.0	0.1	−10.0	0.2
0.0	0.2	−5.0	0.5
5.0	0.5	0.0	0.2
10.0	0.2	5.0	0.1
Hostile			
−20.0	0.2	−5.0	0.1
−10.0	0.25	0.0	0.2
−5.0	0.25	5.0	0.25
0.0	0.2	10.0	0.25
5.0	0.1	20.0	0.2

4. SIMULATION

To evaluate the performance of the POMDP policies, we used the Collision Avoidance System Safety Assessment Tool (CASSATT), developed at Lincoln Laboratory. CASSATT performs fast-time Monte Carlo analysis that takes encounter model data as input and simulates aircraft motion over a period on the order of one minute near the closest point of approach between two or more aircraft. CASSATT has been used for prior TCAS safety analysis [8] and sense-and-avoid development for unmanned aircraft [9].

The CASSATT framework was built in Matlab and Simulink and has been compiled into native code using Real-Time Workshop. The framework was designed to be modular to allow different collision avoidance systems and sensor models to be easily incorporated. As part of this project, we extended CASSATT to allow communication with the collision avoidance system over a TCP/IP socket connection. This extension allows changes to be made to the collision avoidance system without having to recompile the remainder of the CASSATT system. The collision avoidance system runs as a server to which CASSATT connects as a client. Socket communication also allows CASSATT to run on a different machine from the collision avoidance system; for our experimentation however, we always ran the collision avoidance system on the same machine as CASSATT.

Figure 2 provides an overview of the simulation framework. An encounter model generates the initial conditions and scripted maneuvers for both aircraft involved in the encounter. The initial conditions and script are fed into the dynamic simulation model. The sensor model takes as input the current state from the dynamic model and produces an observation, or sensor measurement. The state estimation process updates the belief state based on the observation. The POMDP policy is evaluated on the updated belief state and an optimal, or approximately optimal, action is executed. The dynamic model updates the state, and the process continues until the end of the encounter. The components of the simulation framework are discussed in this section.

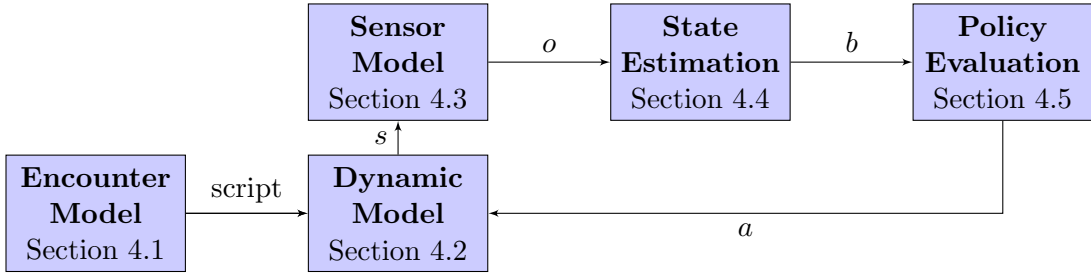


Figure 2. Simulation framework.

4.1 ENCOUNTER MODEL

An encounter model generates the initial conditions and scripted maneuvers for both aircraft involved in the encounter. Several encounter models have been incorporated into CASSATT and have been used for a variety of past analyses. We used the encounter model developed by Lincoln Laboratory for cooperative aircraft [10]. This encounter model was developed from 9 months of National Airspace System radar data from over 120 sensors maintained by the Federal Aviation Administration and Department of Defense. A dynamic Bayesian network representing the behavior of the aircraft was learned from actual encounters extracted from the dataset. Generating new encounters for use in Monte Carlo analysis involves sampling from this dynamic Bayesian network.

4.2 DYNAMIC MODEL

Aircraft dynamics are represented using a tunable 6 degree-of-freedom, point-mass dynamic model, which includes aircraft transient response characteristics and performance limits such as maximum pitch rate or bank angle. In general, the aircraft in CASSATT fly trajectories that are defined by an encounter model and based on aircraft turn rate, vertical rate, and airspeed acceleration. These control values may change every tenth of a second.

4.3 SENSOR MODEL

We implemented models for four types of sensors. Our sensor models include sensor specifications, the simulation of sensor hardware to compute sensor readings, and further processing of the sensor readings to convert them into the POMDP formulation that operate on the Relative Coordinate System.

Section 3 introduced the Relative Coordinate System. There are three other coordinate systems that are used by the sensor models:

- **Global coordinate system:** This is the global coordinate system, also known as the earth coordinate system, used by the CASSATT simulation framework. The origin is an arbitrary point chosen by CASSATT. Positive x is east, positive y is north, and positive z is altitude.
- **Local coordinate system:** This is the egocentric local coordinate system of the aircraft. The origin is the aircraft center of mass. Positive x is in the direction of the right wing, positive y is the direction of the nose, and positive z is upwards.
- **Auxiliary coordinate system:** This is an egocentric local coordinate system whose x - y - z axes are aligned with the east-north-altitude axes of the global coordinate system. The origin is the aircraft center of mass. Positive x is in the direction of the east axis of the global coordinate system. Positive y is in the direction of north axis of global coordinate system. Positive z is in the direction of altitude axis of global coordinate system.

The pseudocode for simulating sensor readings for all types of sensors is given in Appendix A as Algorithm 1. To process measurements from the TCAS, radar, and perfect sensors, we find the coordinates of the intruder in the auxiliary coordinate system and convert those coordinates into vertical and horizontal distances from our aircraft to the intruder. To process measurements from the EO/IR sensor, we find the coordinates of a unit line-of-sight vector in the auxiliary coordinate system and convert those coordinates into bearing and elevation values in the auxiliary coordinate system. Figure 3 summarizes the sensor capabilities, and Table 4 summarizes the sensor specification parameters.

4.4 STATE ESTIMATION

Input to the state estimation (or belief-state update) process consists of the following:

- A POMDP formulation
- A belief-state (probability distribution over the state set that reflects our belief about the true state)
- The action executed
- The observation produced as a result of taking that action

Classical belief-state updating is a two-step process:

1. Using the transition model of the POMDP formulation, we iterate over all states (possible start states) in our belief-state, and compute which states we might land in (possible end states) as a result of taking the given action. We also compute the probability of landing in each of the possible end states. The output of this step is an intermediate belief-state (a probability distribution over the end states).

	TCAS	Radar	EO/IR	Perfect
Range	■	■		
Bearing	■	■	■	
Altitude	■	■		■
Elevation		■	■	
Range rate		■		
Line-of-sight rate			■	
Position				■

Figure 3. Sensor capabilities. Green, yellow, and red indicate good, moderate, and poor performance, respectively.

TABLE 4
Specification parameters.

TCAS		
Range	5	NM
Altitude quantization	25	ft
Range error standard deviation	50	ft
Bearing error standard deviation	10	deg
Altimetry error scale	40	real
False positive measurement probability	0.00	real
False negative measurement probability	0.01	real
Radar		
Range	5	NM
Minimum azimuth	-110	deg
Maximum azimuth	110	deg
Minimum elevation	-15	deg
Maximum elevation	15	deg
Range error standard deviation	50	ft
Bearing error standard deviation	1	deg
Elevation error standard deviation	1	deg
Range rate error standard deviation	10	ft/s
False positive measurement probability	0.01	real
False negative measurement probability	0.01	real
EO/IR		
Range	5	NM
Minimum azimuth	-110	deg
Maximum azimuth	110	deg
Minimum elevation	-15	deg
Maximum elevation	15	deg
Bearing error standard deviation	0.5	deg
Elevation error standard deviation	0.5	deg
Line-of-sight rate error standard deviation	0.5	deg/s
False positive measurement probability	0.01	real
False negative measurement probability	0.01	real
Perfect		
Range	5	NM

2. According to the observation model of the POMDP formulation, some of the end states might not be consistent with the given observation. We filter out those end states (eliminate them from our intermediate belief-state), and also adjust the probabilities of the remaining end states according to the observation model. The output of this step is the final updated belief-state.

Practical implementations of belief-state update process might use cascaded iterations over state and observation sets, and carry out both steps at once without generating an actual intermediate belief-state. Another important practical aspect of the belief-state update process is the overall computation time. Most applications that use POMDP models need to maintain an up-to-date belief-state, and therefore the state estimation is one of the key components of such applications that should be executed frequently. The computation time becomes even more crucial in real-time applications such as our collision avoidance system. In such cases, it is customary to make use of special data structures such as sparse matrices to represent large transition and observation models in order to be able to quickly skip over elements with zero probabilities to speed up computation time.

One contribution of this project is the development of a novel method that drastically reduces the computation time of the belief-state update process. The method involves the following components:

- **An extended transition model:** We combine the transition and observation models of a POMDP formulation into a single model that we call an extended transition model, represented as a large lookup table. Given a start state, action, and observation, the table provides a list of end states we can land in and their probabilities. We developed a software package (called POMDP Processor) that reads in a POMDP formulation, computes the extended transition table, and outputs it in the form of a new POMDP formulation that has a state set, an action set, an observation set and an extended transition model. We call this new POMDP formulation a Processed POMDP, or PPOMDP for short. A PPOMDP file is usually larger in size than a POMDP file. PPOMDPs can be computed offline, and their main purpose is time efficiency during belief-state updates rather than memory or storage efficiency.
- **A special data structure for representing belief-states:** Similar to PPOMDPs, this data structure is designed with time-efficient computation in mind. It occupies more than four times the size of a naïve belief-state representation in the memory. All of the required space is allocated at once during initialization, and no further dynamic allocation or deallocation is performed. It uses a sparse representation to easily store, fetch and iterate over only the states with non-zero probabilities. It uses the pointer data type (in the C Programming Language) to switch between arrays containing various data and thereby it never requires resetting an array, or copying array values from one place to another in memory. It also keeps running sums of probabilities; therefore it also never requires summing over array elements.

In our tests, we witnessed speed-ups in belief-state updates up to factors of 100 to 1000. The strength of the PPOMDP formulation comes from the fact that many of the combinations (start state, action, and observation triplets) do not lead to any end states. We call such combinations

TABLE 5
POMDP processing for the TCAS sensor.

Number of states in the POMDP formulation	2886
Number of actions in the POMDP formulation	3
Number of observations in the POMDP formulation	183
Total number of (state-action-observation) combinations	1584414
Number of combinations with no end states (vanish)	1205140
Number of combinations with at least one end state (persist)	379274
Total number of end states reachable from all combinations	8140937
Maximum number of end states reached from a single combination	962
Average number of end states reached from a single combination	5.13814
Time spent computing the PPOMDP formulation (mm:ss)	07:29

vanishing combinations. For the rest of the combinations, called *persisting combinations*, it is usually the case that the number of end states is only a very small fraction of the whole state set. Here, we provide two examples.

The first example, summarized in Table 5, demonstrates a general reduction pattern observed in our tests. This example is from a POMDP formulation for the TCAS sensor using ± 1500 ft/sec vertical velocity limits. The processing was done on an Intel Core 2 Duo CPU running at 2.5 GHz, with 4 GB available system memory. Table 6 contains a histogram showing the frequency of number of end states for the TCAS sensor. When we use the novel belief-state representation and the PPOMDP model, we only iterate over the states with non-zero probabilities in the belief-state, and for each such state, we only consider 5 or 6 end states on average.

Our second example, summarized in Table 7, demonstrates the efficiency of the PPOMDP formulation. This example is from a POMDP formulation for the perfect sensor using ± 1500 ft/sec vertical velocity limits. In the POMDP formulation for the hypothetical perfect sensor, the observations are very accurate and they help filter out irrelevant end states very effectively. In a sense, the perfect sensor POMDP model is close to an MDP model. Table 8 contains a histogram showing the frequency of number of end states for the perfect sensor.

4.5 POLICY EVALUATION

For large POMDPs, solvers (e.g., SARSOP or HSVI2) may require hours or days to find policies with tight regret bounds. Most solvers usually generate a simple solution first (probably with loose regret bounds), and iteratively improve that solution until the user interrupts the process. Policy evaluation involves determining the action recommended by a policy for the current belief state.

TABLE 6**Histogram showing the frequency of number of end states for the TCAS sensor.**

End states	Frequency	End states	Frequency
0	1205140	28	100
1	17	29	1316
2	1630	30	17850
3	1210	31	582
4	18090	32	11152
5	1048	33	64
6	22500	36	25014
7	82	37	754
8	28480	40	5376
9	6982	41	160
10	16250	43	4
11	896	44	1016
12	54480	45	6564
13	746	46	256
14	320	47	996
15	9550	48	12000
16	22934	49	516
17	1074	54	6560
18	31160	55	480
19	404	64	2442
20	14490	70	16
21	466	71	896
22	120	72	4768
23	80	144	170
24	36784	192	170
25	4644	240	340
26	88	288	170
27	5000	962	17

TABLE 7**POMDP processing for the perfect sensor.**

Number of states in the POMDP formulation	2886
Number of actions in the POMDP formulation	3
Number of observations in the POMDP formulation	183
Total number of (state-action-observation) combinations	1584414
Number of combinations with no end states (vanish)	1480336
Number of combinations with at least one end state (persist)	104078
Total number of end states reachable from all combinations	581187
Maximum number of end states reached from a single combination	16
Average number of end states reached from a single combination	0.366815
Time spent computing the PPOMDP formulation (mm:ss)	07:11

TABLE 8**Histogram showing the frequency of number of end states for the perfect sensor.**

End states	Frequency
0	1480336
1	7057
2	2701
3	2500
4	27076
5	256
6	45296
9	18172
16	1020

The space of all belief-states for a given POMDP formulation is called the belief simplex. An optimal policy maps belief-states (which corresponds to a point in the belief simplex) to actions that maximize expected long-term reward. Due to the continuous nature of the belief simplex, policies are usually represented as a collection of regions of belief simplex, and the associated actions that should be taken when the given belief-state falls inside those regions. More specifically, a policy consists of a set of α -vectors and there is an action associated with each α -vector. An α -vector serves two purposes:

1. The maximum cardinality for an α -vector is the number of states in the POMDP formulation. Usually, most of the α -vectors of a policy contain less than the maximum number of entries. The entries that are present in an α -vector determine the region of the belief simplex to which the α -vector is applicable. Usually a policy contains a single α -vector or a small number of α -vectors with maximum cardinality. These vectors are applicable to all of the belief simplex (all possible belief-states). The rest of the α -vectors in the policy are specialized to different sub-regions. Note that different sub-regions might overlap, and they are not required to cover the belief simplex. Also, a policy may contain multiple α -vectors that are applicable to the same region.
2. The inner product of an α -vector and the belief-state yields the expected long-term reward in the case of taking the action associated with that α -vector.

As a result, the algorithm for policy evaluation takes the following form:

- Determine the applicable α -vectors for the given belief-state.
- Compute expected long-term rewards by taking inner products of all applicable α -vectors with the given belief-state.
- The best action is the one that is associated with the α -vector that yields the highest expected long-term reward.

Since policy evaluation is also executed frequently similar to the state estimation process, we implemented a time-efficient data structure for working with policies. Our design leverages the special data structure for belief-states, and allows us to quickly compute the best action.

This page intentionally left blank.

5. RESULTS

Monte Carlo safety studies of collision avoidance systems generally involve exposing a collision avoidance system to a collection of encounters selected from some distribution $p(x)$. If $f(x)$ is the probability encounter x leads to an NMAC and x_1, \dots, x_N are encounters chosen independently from $p(x)$, then the probability of an NMAC may be estimated as follows:

$$\Pr(\text{NMAC}) = \frac{1}{N} \sum f(x_i).$$

To test our POMDP policies, we used the encounter model developed by Lincoln Laboratory for cooperative aircraft [10]. Most of the encounters generated by the encounter model do not result in an NMAC. We would need to sample from the $p(x)$ encoded by the model many times before generating a test case that results in an NMAC. To reduce the number of samples required before we generate an “interesting” encounter, we sample from an alternative distribution $q(x)$ that focuses on encounters with low vertical and horizontal miss distances at the time of closest approach. Because we are no longer sampling from $p(x)$ we need to weight the samples in order to produce an accurate estimate of $\Pr(\text{NMAC})$:

$$\Pr(\text{NMAC}) = \frac{1}{N} \sum f(x_i)p(x_i)/q(x_i).$$

This approach is known as *importance sampling* and results in a better estimate of $\Pr(\text{NMAC})$ using fewer samples.

We generated 15,000 encounters from the encounter model using importance sampling. Approximately 15,000 samples are necessary to provide a reasonable estimate of $\Pr(\text{NMAC})$. In the future, we would like to test our system using at least hundreds of thousands of samples to provide better performance estimates. Because the encounters may be simulated in parallel, we used the parallel computing environment at Lincoln Laboratory, known as LLGrid. Using 64 compute nodes, it takes approximately 10 minutes to evaluate a POMDP policy on 15,000 encounters.

Although evaluation is rather fast, SARSOP requires at least an hour to generate satisfactory policies for a POMDP specification. The development process was hindered, to some extent, by the time required to find acceptable policies for a given model. Many of the results presented in this section would be improved if more time was allotted to the POMDP solver. The regret bounds after 1 to 3 hours are still quite loose. We have experienced some difficulty with SARSOP when running for more than a few hours on certain large POMDPs. The authors of SARSOP are currently working on improving their memory management routines, and we expect better policies in the future.

Table 9 summarizes some preliminary results of the POMDP policies on 15,000 encounters. For these experiments, the magnitude of the vertical rate is limited to 2500 ft/min. For comparison, we provide results from the Basic CAS strategy. Basic CAS uses the same sensor models as the POMDP system, but it simply accelerates up with 0.25 g if an intruder is seen below and accelerates down with 0.25 g if an intruder is seen above. The table shows the risk ratio for different combinations of sensors and algorithms. The risk ratio associated with a particular system is the probability that an encounter leads to an NMAC using the system divided by the probability that

TABLE 9

Risk ratio for different sensors and aircraft performance constraints.

	POMDP CAS			Basic CAS		
	ratio	vel.	accel.	ratio	vel.	accel.
TCAS	0.00277	14.13303	1.75919	< 0.00001	32.90970	1.03470
Radar	0.06337	23.62831	1.26154	0.05083	19.23245	2.40971
EO/IR	0.03510	28.61076	1.47691	0.04724	19.45035	2.30833
Perfect	0.00117	18.68779	1.09518	< 0.00001	33.03076	0.79019

an encounter leads to an NMAC without the system. Of course, better performance is indicated by a small risk ratio. Table 9 also shows the mean vertical velocity magnitude in ft/s and mean vertical acceleration magnitude in ft/s². It is desirable to have these values as small as possible without sacrificing the risk ratio. Large values of mean velocity magnitude and mean acceleration magnitude indicate that the aircraft is maneuvering unnecessarily.

Table 9 shows that the POMDP policies significantly reduce risk for all sensor modalities. The Basic CAS reduces risk even further (except for EO/IR sensor), but maneuvers more frequently, as we would expect. For comparison, we ran TCAS version 7 and obtained a risk ratio of 0.06122. The POMDP policy for the TCAS sensor is about 20 times safer than the TCAS version 7 currently used on manned aircraft. However, TCAS has a much lower mean vertical velocity magnitude (5.09436 ft/s), indicating that it maneuvers less frequently.

Although we used the same sensor model for the TCAS algorithm as the TCAS POMDP and constrained the vertical rate magnitude to be within 2500 ft/min, the comparison is not entirely fair. TCAS was designed for pilot-in-the-loop control and assumes a delay between when the resolution advisory is issued and when the pilot responds. Although the POMDP algorithm has the advantage over the TCAS algorithm because it can maneuver instantaneously, the TCAS algorithm is permitted to make up to 0.35 g maneuvers whereas the POMDP was constrained to 0.25 g maneuvers. CASSATT models the pilot response to TCAS resolution advisories as being 0.25 g following a 5 s delay for the initial advisory and then 0.35 g with a 2.5 s delay for subsequent advisories. Although a direct comparison between the TCAS POMDP and TCAS algorithm cannot be made, we can be confident, at least, that the TCAS POMDP is performing well.

We see that the radar and EO/IR sensors have higher risk of collision than the TCAS and perfect sensors. This is because their performance is inherently limited by their field-of-view constraints. However, we believe that performance could further be improved with a better POMDP model that could accurately predict the field of view of the aircraft. Addressing this issue will require an extension to the state space. This will be the subject of follow-on work.

It must be emphasized that the relative cost of maneuvering and colliding was chosen arbitrarily for these results. If we wanted to increase safety at the expense of maneuvering more

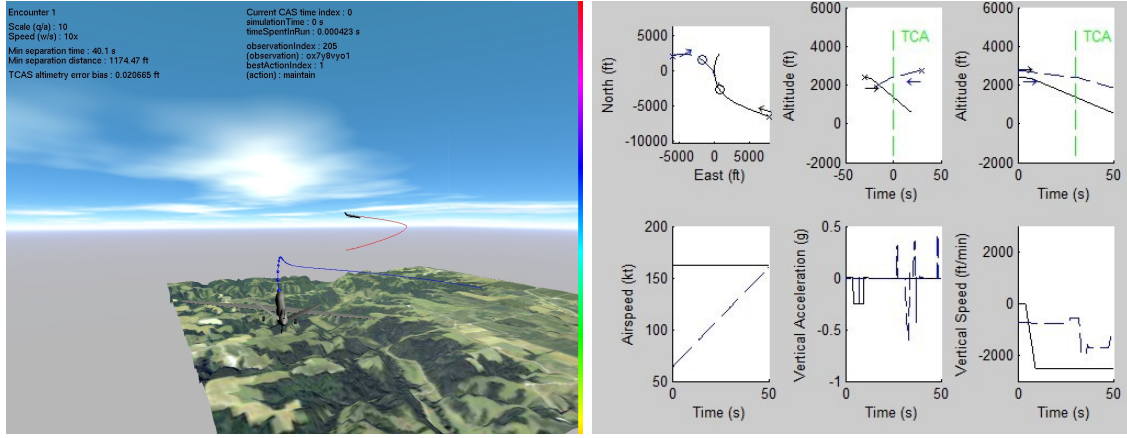


Figure 4. A screenshot of the Encounter Analyzer showing flight trajectory from own aircraft point of view, using the TCAS sensor and POMDP policy, as well as an encounter summary from CASSATT.

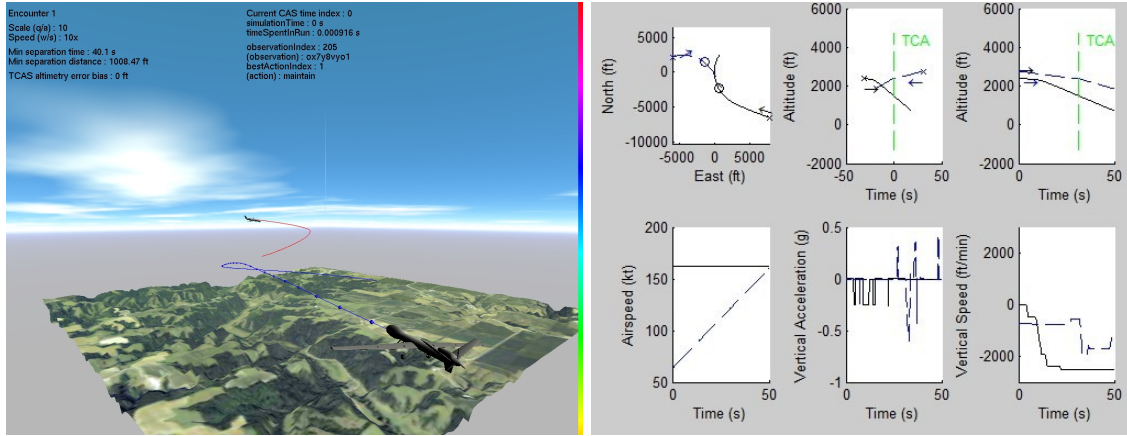


Figure 5. A screenshot of the Encounter Analyzer showing flight trajectory from own aircraft point of view, using the radar sensor and POMDP policy, as well as an encounter summary from CASSATT.

frequently, we can simply change our cost function and rerun SARSOP to generate a new policy. Further work will involve varying the relative cost to trace out the system operating characteristic (SOC) curve [11, 12].

Figures 4, 5, and 6 show sample encounters with the three principal sensor models. The left pane in each figure shows the trajectory of our aircraft in blue and that of the intruder in red. On the right pane are plots produced by CASSATT.

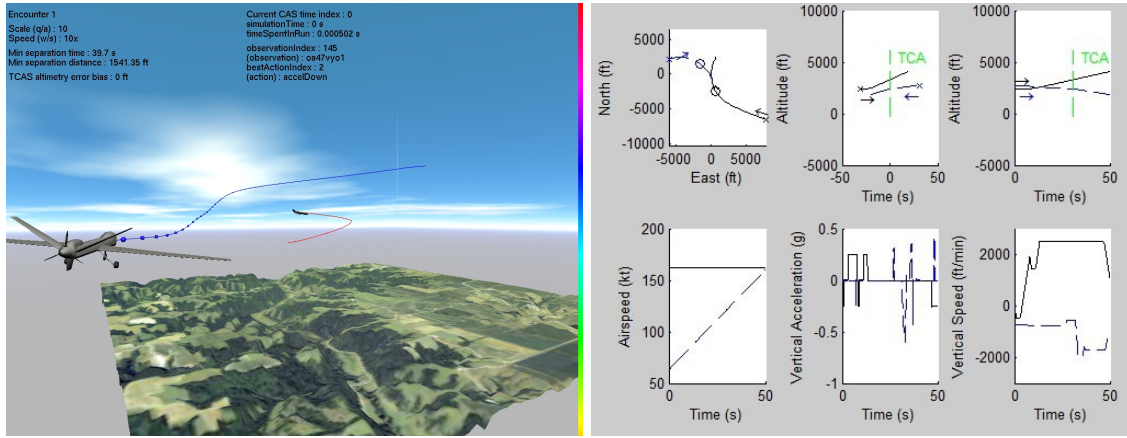


Figure 6. A screenshot of the Encounter Analyzer showing flight trajectory from own aircraft point of view, using the EO/IR sensor and POMDP policy, as well as an encounter summary from CASSATT.

6. DISCUSSION

This section discusses the limitations of our initial POMDP models and identifies areas for further improvement. In addition, this section discusses some issues identified by examining failure modes in simulation.

6.1 DISCRETIZATION

Our state space representation captures most of the features that are necessary in selection of an action to avoid collisions, but there is a loss of information when we go from two 13-dimensional aircraft state vectors to a 5-dimensional state space. One way to improve performance is to augment the POMDP state space with more features from the underlying true state space. Another way to improve performance is to use a finer grained discretization, which involves adding more bins along each dimension. Both of those approaches cause exponential growth in the size of the state space and the time it takes to compute the policy. For example, generating a reasonable policy for a POMDP with 3000 states requires approximately half an hour. Adding a couple bins to each dimension increases the number of states to over 9000 and the time to generate a reasonable policy increases to over five hours.

6.2 PARAMETER VALUES

Our POMDP models contain many parameters that have not been tuned to the encounter model. Many of the parameters were chosen without any particular justification. We believe that performance can be significantly improved by better matching the internal model used for decision making to the encounter model. As an example, the EO/IR Extended Observation Function described in Algorithm 4 uses an internal precautionary *angular margin* parameter when converting a rectangular bin into two angle values (the maximum and minimum angles that “see” the bin). Tuning this parameter reduces the EO/IR risk ratio to 0.03510 down from 0.11038.

6.3 MISSING STATE INFORMATION

There are features that may improve performance that are currently not part of our state space. One important feature is our roll angle. With limited field-of-view sensors such as the radar and EO/IR sensors, sometimes whether the intruder falls into the active angular range of the sensor or not depends on how much we are banking. In the current formulation, there is no way to estimate the current roll angle from a given state, making it impossible to determine intruder detectability by projecting the active angular range of the sensors onto the 2-D plane. We currently assume a fixed zero-degree roll angle and add some precautionary margins, but this affects the performance in two ways:

- If the roll angle is actually zero, non-zero probability would be assigned to some undetectable bins that are inside the margins.

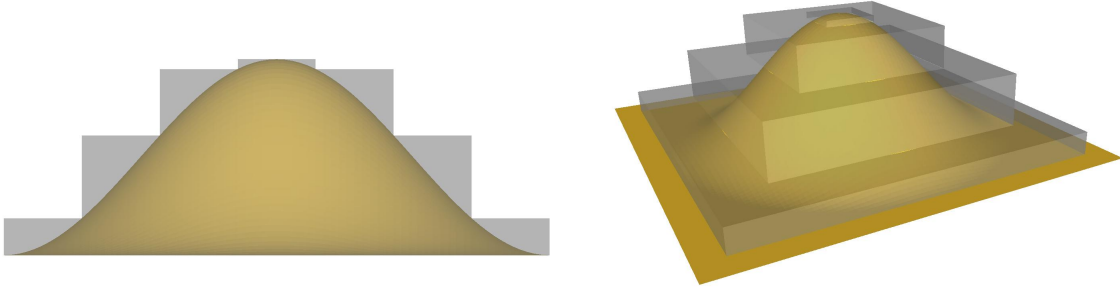


Figure 7. Gaussian distribution approximated by four flat distributions stacked on top of each other.

- If the roll angle is larger than the margins and the intruder is detectable, zero probability would be assigned to a case that is actually possible. This results in a belief state crash, which is a belief update that results in zero probability assigned to all states.

Another important feature in computing active sensor range is pitch angle. Pitch angle can be estimated from our vertical and horizontal velocities. The vertical velocity information is present in a given state (in the form of a bin, which is therefore a range rather than a single value). In our POMDP models, we use the minimum (stall) and maximum airspeed values for own aircraft type and estimate a lower and upper bound for the pitch angle.

6.4 OBSERVATION MODELS

The error models for most of the sensor measurements are Gaussian, but we initially used piecewise-uniform distributions to simplify implementation in our discrete state space. Such a flat distribution might be an acceptable approximation for positional sensors like TCAS, but it is not appropriate when the error is large as in angular sensors such as radar and EO/IR. Flat probability distributions over large regions do not help the POMDP model localize the intruder in the projected 2-D plane. When it was determined that performance was negatively affected by the observation model, we implemented a method to coarsely discretize a Gaussian distribution as shown in Figure 7 and apply it to 2-D bins. Using the discretized Gaussian distribution resulted in a 10 times better risk ratio for the radar sensor (bringing it down from 0.67659 to 0.06337). We believe that a better Gaussian discretization scheme or an analytical solution would further improve results.

6.5 ESTIMATION OF INTRUDER VERTICAL VELOCITY

Similar to TCAS, our current models adjust the vertical speed of the aircraft to avoid collision. Hence, it is important that the vertical speed estimates be as accurate as possible. Unfortunately, this requires a much finer discretization of the heights of the 2-D bins in the projected plane, which in turn results in a significant increase in the number of states. The problem with a coarse discretization is shown in Figure 8.

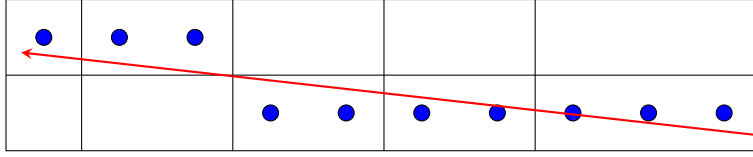


Figure 8. Path of intruder and discretized observations.

The 2-D bins of the projected aircraft are drawn in black. The red line shows the projected (relative) path of the intruder. Blue dots are the points in time when we make observations. The blue dots are drawn in the respective bins that are perceived as the observations. As can be seen in the figure, the first seven observations are all from the lower bins, and therefore the POMDP model assigns a high probability to vertical velocity values that are close to zero. With the eighth observation, the intruder enters one of the upper bins, resulting in a high probability being assigned to high vertical velocity values. This causes us to overestimate the intruder vertical rate.

6.6 ESTIMATION OF CLOSURE RATE

In order to keep the state space small and still be able to cover a very large 2-D projected region, we used variable-sized bins and discretized the space near the origin more finely. There is however an issue with wide bins that we did not anticipate before running our experiments. As Figure 8 shows, a wide bin means that we will be getting the same observation repeatedly until the projection of the intruder falls into another bin in the 2-D projected plane. This affects the relative horizontal velocity (closure rate) estimation. Successively receiving the same observation creates the illusion of a stationary intruder in the horizontal plane, and suddenly receiving a different observation results in excessively high velocity estimates. As in the vertical velocity case, a finer discretization is required to alleviate this problem.

6.7 THREE-DIMENSIONAL TRAJECTORY EXTRAPOLATION

Some of the problems with the current POMDP models are due to the 2-D projection of the underlying 3-D space, but we hypothesized that many of these problems can be solved by estimating motion in full 3-D coordinates. To test our hypothesis, we implemented a simple, non-POMDP collision avoidance system, which we call Analytic CAS because of the way it analytically propagates aircraft trajectories in 3-D. Analytic CAS is suited for sensors that provide full positional information (as opposed to, for example, the EO/IR sensor, which provides only angular information). The algorithm used by Analytic CAS is very simple: it collects 3-D position data for both aircraft and estimates velocities and accelerations (also in 3-D) by simple differentiation. We implemented velocity and acceleration estimators for the perfect and TCAS sensors. Using simple quadratic equations, it determines whether the aircraft will have a close encounter. We implemented two different versions for the close-encounter test. In the first version, a close encounter is defined using a 3-D cylinder. In the second version, a close encounter occurs when the vertical separation drops below some threshold. If a close encounter is expected, an evasive maneuver is performed

TABLE 10**Risk ratios for analytic collision avoidance system.**

	Analytic CAS			Basic CAS		
	ratio	vel.	accel.	ratio	vel.	accel.
Perfect sensor, 3-D	0.05456	4.56433	0.22473	< 0.00001	33.03076	0.79019
Perfect sensor, altitude	0.01697	5.59747	0.76899	< 0.00001	33.03076	0.79019
TCAS sensor, 3-D	0.08010	7.40275	1.09657	< 0.00001	32.90970	1.03470
TCAS sensor, altitude	0.02050	19.55749	4.51164	< 0.00001	32.90970	1.03470

immediately to increase the altitude by 200 feet. The risk ratios for Analytic CAS are shown in Table 10. The relatively good results produced by such a simple strategy suggests that moving our models and computations to 3-D might significantly improve performance.

7. CONCLUSIONS

We have learned a number of lessons from this initial investigation of the application of the POMDP framework to unmanned aircraft collision avoidance.

- The POMDP formulation is flexible enough to accommodate a variety of sensor modalities, intruder behavior, aircraft dynamics, and cost functions.
- Complex policies produced by POMDP solvers can be readily implemented in real time. Both state estimation and policy execution are quite efficient for the state spaces considered in this research.
- Current state-of-the-art POMDP solvers, using a simplified representation of the aircraft dynamics, can generate useful collision avoidance behavior.

However, there are also some clear challenges that need to be addressed as we move forward:

- Improvements in formulation and in the solvers will likely be needed to increase the fidelity of our models. In particular, we have limited our formulation to representing motions in two (relative) dimensions. Moving to full three-dimensional motion in a discretized formulation will take the size of the state space beyond the range of existing solvers. We need to investigate other representations for the state space and new types of solvers that suffer less from exponential explosion than pure discretized representations.
- There are many parameters in our models; they capture properties of aircraft dynamics, of sensors, encounters, etc. At the moment, these parameters were chosen by hand. There is a wealth of available data that should be used to select these parameters automatically.
- There is a need for more sophisticated tools to support understanding and debugging of the policies produced by solving the POMDPs. The Encounter Analyzer has been an important step in this direction but additional development is necessary.

The POMDP approach holds much promise for implementing collision avoidance systems that involve a wide range of aircraft type and sensor suites.

This page intentionally left blank.

8. FUTURE WORK

TCAS is currently mandated onboard all large transport aircraft in the world, but the logic will need to be adapted as airspace procedures and surveillance technologies evolve over the next 20 years. A POMDP approach would aid in the development of new systems. Instead of manually constructing and tuning another complex TCAS algorithm, the focus would be on the much easier task of developing accurate models of flight dynamics, intruder behavior, and sensor systems. Given a cost function, an automated POMDP solver would be assigned the difficult task of optimizing the logic. Of course, the resulting logic would still be required to undergo rigorous analysis and flight tests before certification, but the POMDP approach would likely speed development.

It is uncertain whether a policy represented as a collection of α -vectors ought to be used directly in the aircraft avionics. In theory, one could upload a set of α -vectors that has been optimized for their particular aircraft and sensor package. Upgrading the logic as the airspace operational environment evolves would be a matter of inputting a new set of α -vectors. One disadvantage of using a set of α -vectors directly is that it is difficult for a human to understand the strategy it encodes. One advantage with logic written in pseudocode is that a human can step through the logic and gain some understanding of how it works. However, the logic that TCAS follows is quite complex in some situations, such as a multi-intruder encounter, that even the pseudocode can be difficult to understand. Further research may involve investigating methods to automatically convert policies represented as α -vectors to more human-readable representations.

The current TCAS sensor relies entirely on radar beacon measurements, but the next generation of TCAS may also use Automatic Dependent Surveillance-Broadcast (ADS-B), which uses signals from the Global Navigation Satellite System. ADS-B would provide significantly better position and velocity estimates as well as intent information. However, there is concern about relying solely on the Global Navigation Satellite System for separation because of the potential for blackouts. A collision avoidance system that uses both ADS-B position and intent information and traditional TCAS sensor measurements would be more robust to failure in either system. A POMDP approach, using a model of sensor accuracy, reliability, and availability, could help inform how best to fuse ADS-B and TCAS sensor data.

Whether or not a policy obtained from a POMDP solver is ultimately used in the next generation of TCAS, it can serve as a baseline to which a human-designed logic can aspire. A POMDP framework could help answer a number of questions that are important to next generation TCAS development. For example, the framework could help quantify the benefit of modeling pilot response uncertainty. This project has focused on collision avoidance with automated response, but the next generation of TCAS will probably require human pilot response. By modeling the delay of the human response and the degree to which the response follows the TCAS resolution advisory, it is conceivable that a more robust collision avoidance system could be developed. Interest in ensuring that TCAS is robust to pilots not properly following resolution advisories has increased after the midair collision over Überlingen in 2002 where one pilot followed Air Traffic Control instead of TCAS.

For both manned and unmanned aircraft, there is interest in investigating the use of horizontal avoidance maneuvers. Instead of trying to avoid collision simply by adjusting the vertical rate, as is currently done in TCAS and in this project, it may be more effective to turn. Adding additional control variables and extending the current state space to explicitly model three-dimensional dynamics will significantly increase the complexity of the POMDP representation. Further research would involve investigating alternative solution methods for POMDPs.

This project has focused on discrete-time POMDPs with finite states, actions, and observations. Moving to a continuous-time representation with continuous states, actions, and observations may result in better policies. Handling high-dimensional state spaces that include other variables not included in the unmanned aircraft model, such as ADS-B intent information, may be challenging for general-purpose discrete solvers. Scaling to more sophisticated state-transition models is likely to require leveraging the structure inherent in collision avoidance problems.

APPENDIX A

PSEUDOCODE

Algorithm 1 Simulation of Sensor Readings

```
Find coordinates of other aircraft in local coordinate system of own aircraft
Compute true readings (values without noise)
if returning a false positive reading then
    return a random reading that complies with sensor specifications
else if returning a false negative reading then
    return no intruders inside sensing range
else
    if other aircraft is outside sensing range then
        return no intruders inside sensing range
    else
        return sensor readings modified according to error model
    end if
end if
```

Algorithm 2 TCAS Extended Observation Function

Input: end state, action, set of all observations

Output: list of probabilities

if end state is a START state, or a DONE state, or is outside sensing range **then**

 Add to the list the single observation that reports only Own-Vy

else

 Locate the bin in the Relative Coordinate System that corresponds to the end state and enlarge the boundaries of that bin by a *margin* determined by the sensor error model

for each observation in the observation set **do**

if the bin that corresponds to the observation overlaps with the enlarged bin **then**

 Add the observation to the list with probability proportional to the overlap area

end if

end for

if the false negative measurement probability of the sensor is greater than zero **then**

 Add the observation that reports only Own-Vy (with false negative measurement probability of the sensor)

end if

end if

Algorithm 3 Radar Extended Observation Function

Input: end state, action, set of all observations

Output: list of probabilities

```
if end state is a START state, or a DONE state, or is outside sensing range then
    if the false positive measurement probability of the sensor is zero then
        Add to the list the single observation that reports only Own-Vy
    else
        Add to the list all observations with equal probability
        Add to the list the single observation reporting only Own-Vy (with probability  $1.0 -$ 
false positive measurement probability)
    end if
else
    Locate the bin in the Relative Coordinate System that corresponds to the end state and
enlarge the boundaries of that bin by a margin determined by the sensor error model
    for each observation in the observation set do
        if the bin that corresponds to the observation overlaps with the enlarged bin then
            Add the observation to the list with probability proportional to the overlap area and
weighted by the sensor error model (discretized Gaussian)
        end if
    end for
    if the false negative measurement probability of the sensor is greater than zero then
        Add the observation that reports only Own-Vy (with false negative measurement proba-
bility of the sensor)
    end if
end if
```

Algorithm 4 EO/IR Extended Observation Function

Input: end state, action, set of all observations

Output: list of probabilities

```
if end state is a START state, or a DONE state, or is outside sensing range then
    if the false positive measurement probability of the sensor is zero then
        Add to the list the single observation that reports only Own-Vy
    else
        Add to the list all observations with equal probability
        Add to the list the single observation reporting only Own-Vy (with probability  $1.0 -$ 
        false positive measurement probability)
    end if
else
    Locate the bin in the Relative Coordinate System that corresponds to the end state and
    determine the minimum and maximum angles that 'sees' this bin
    for each observation in the observation set do
         $d \leftarrow 0$ 
        if the angle that corresponds to the observation is outside the minimum and maximum
        angles that 'sees' the end state then
             $d \leftarrow$  angular distance to the end state
        end if
        Add to the list all observations with probabilities that are proportional to the density at
         $d$  of a Gaussian PDF with a zero mean and standard deviation given by the Elevation Error  $\sigma$ .
    end for
    if the false negative measurement probability of the sensor is greater than zero then
        Add the observation that reports only Own-Vy (with false negative measurement proba-
        bility of the sensor)
    end if
end if
```

Algorithm 5 Perfect Extended Observation Function

Input: end state, action, set of all observations

Output: list of probabilities

```
if end state is a START state, or a DONE state, or is outside sensing range then
    Add to the list the single observation that reports only Own-Vy
else
    Locate the bin in the Relative Coordinate System that corresponds to the end state and add
    to the list the single observation that corresponds to this same bin
end if
```

Algorithm 6 Extended Transition Function

Input: start state, action, set of all states

Output: list of probabilities

Locate Own-Vy bin corresponding to the given start state

Predict new Own-Vy bin boundaries according to the given action (if ‘accelerating up,’ bin boundaries increase by the applied acceleration times ΔT , if ‘maintaining,’ no change occurs, and if ‘accelerating down,’ bin boundaries decrease); add a fixed, small margin when enlarging boundaries

if given start state is a START state **then**

 Add to list the START states that overlap with the predicted Own-Vy bin (with probability proportional to the overlap amount and also scaled according to the probability of staying in START state)

 Add to list all other states that overlap with the predicted Own-Vy bin (with probability proportional to the overlap amount and also scaled according to the probability of appearing in any other state)

else if given start state is a DONE state **then**

 Add to list the DONE states that overlap with the predicted Own-Vy bin (with probability proportional to the overlap amount)

else

 Locate all the bins (X, Y, Other-RelativeVx, Other-Vy, Own-Vy) corresponding to the given start state

for all possible values of intruder horizontal and vertical acceleration model values **do**

 Predict new state (new bin boundaries) using dynamics equations, ΔT , and performance limits of our aircraft

 Add to list all the states that overlap with the predicted state (with probability proportional to the overlap amount)

end for

end if

This page intentionally left blank.

APPENDIX B

POMDP GENERATION

We implemented a software application (called POMDP Generator) that generates POMDP formulations for all four types of sensors. We have already given the pseudocode for all of the algorithms used by the POMDP Generator in previous sections (Reward, Observation and State-Transition Functions) and we have also described the important parameters of those algorithms. Therefore, in this section we will briefly go over the POMDP generation process.

POMDP Generator reads in a couple of text files (specification files) that contain values of various parameters. These specification files are easily editable, and different POMDPs can be generated using different configurations of the parameter values. Collectively, the following data are required and gathered from the specification files:

- Specifications for the respective sensor (given in Table 4)
- Controller frequency (ΔT)
- Aircraft dynamic model (vertical velocity limits, and vertical acceleration magnitude)
- Size of the desired protected space around own aircraft
- State space
- Action space (this is fixed for all sensor types: accelerate up, maintain, accelerate down)
- Observation space (for some sensor types, this is derived from the state space)
- Reward model (crash cost, protected space cost and vertical velocity cost)
- Transition probabilities from START state of the POMDP formulation
- Vertical and horizontal acceleration models for the intruder

After gathering necessary information, the following tasks are performed in the given order:

1. Number of states, actions and observations are determined using the specifications of the state, action and observation spaces. Symbolic names for all states, actions and observations are computed and recorded as part of the POMDP formulation. (Most of the algorithms use integer indices when dealing with states, actions and observations, but there are some algorithms that make use of symbolic names to quickly extract information about a given state, action or observation. Also the symbolic names are good for debugging purposes.)
2. Initial belief-state is computed and recorded. For all sensor types, initial belief-state contains a uniform probability distribution over the duplicated START states, but the number of duplicated START states depends on the state space (more specifically, the number of bins in the discretization of Own- V_y values).

3. POMDP Generator iterates over all state-action pairs and invokes the extended transition function (Algorithm 6) for each pair to compute and record the transition model.
4. POMDP Generator iterates over all states and invokes the relevant extended observation function (one of Algorithms 2, 3, 4, or 5) for each state to compute and record the observation model.
5. For each state, reward function is invoked and reward model is recorded.

The output of the POMDP Generator is a POMDP file. The POMDP files we generated and used in our tests have sizes ranging between 45–55 MB.

APPENDIX C

ENCOUNTER ANALYZER

The data flow between different modules of our development system is as follows:

- CASSATT simulations run at 10 Hz. At each simulation step CASSATT computes aircraft state vectors for both own and intruder aircraft. Since each encounter usually lasts 50 seconds, this corresponds to a total of 1000 aircraft state vectors.
- The software module that implements our collision avoidance algorithms is called the “controller.” CASSATT invokes the controller at 1 Hz (at every 10th simulation step). The inputs to the controller are state vectors for both aircraft and an control command for own aircraft. The control command consists of a vertical rate (or a vertical acceleration), a turn rate, and airspeed acceleration. The aircraft control command provided by CASSATT is the scripted maneuver for our aircraft. It represents the Air Traffic Control (ATC) command that should be followed if there is no danger of collision.
- The output from the controller is another aircraft control command. This command might be a replica of the ATC command, or it might be a different one as a result of the planned collision avoidance maneuver by the controller. This output is captured by CASSATT and used to calculate our state vector for the following 10 simulation steps (before the controller is invoked again).
- At the end of each encounter, CASSATT outputs the horizontal and vertical miss distances (HMD and VMD) at the time of closest approach. An near mid-air collision (NMAC) occurred if HMD is less than 500 ft and VMD is less than 100 ft.

At each invocation, the controller processes its inputs and generates the following internal data to compute its output:

- A sensor reading is simulated (computed) using the state vectors for both aircraft.
- The simulated sensor reading is further processed and converted to an “observation” suitable for the POMDP used by the controller.
- The controller keeps a record of the previous collision avoidance action it took. It also maintains an internal belief-state. At each invocation, this belief-state is updated (using the previous action and the computed observation). Then, using this updated belief-state, the controller decides what collision avoidance action to take next (which determines the output aircraft control command).
- Statistical data is collected inside the controller (such as the total and average times it takes to process all data).

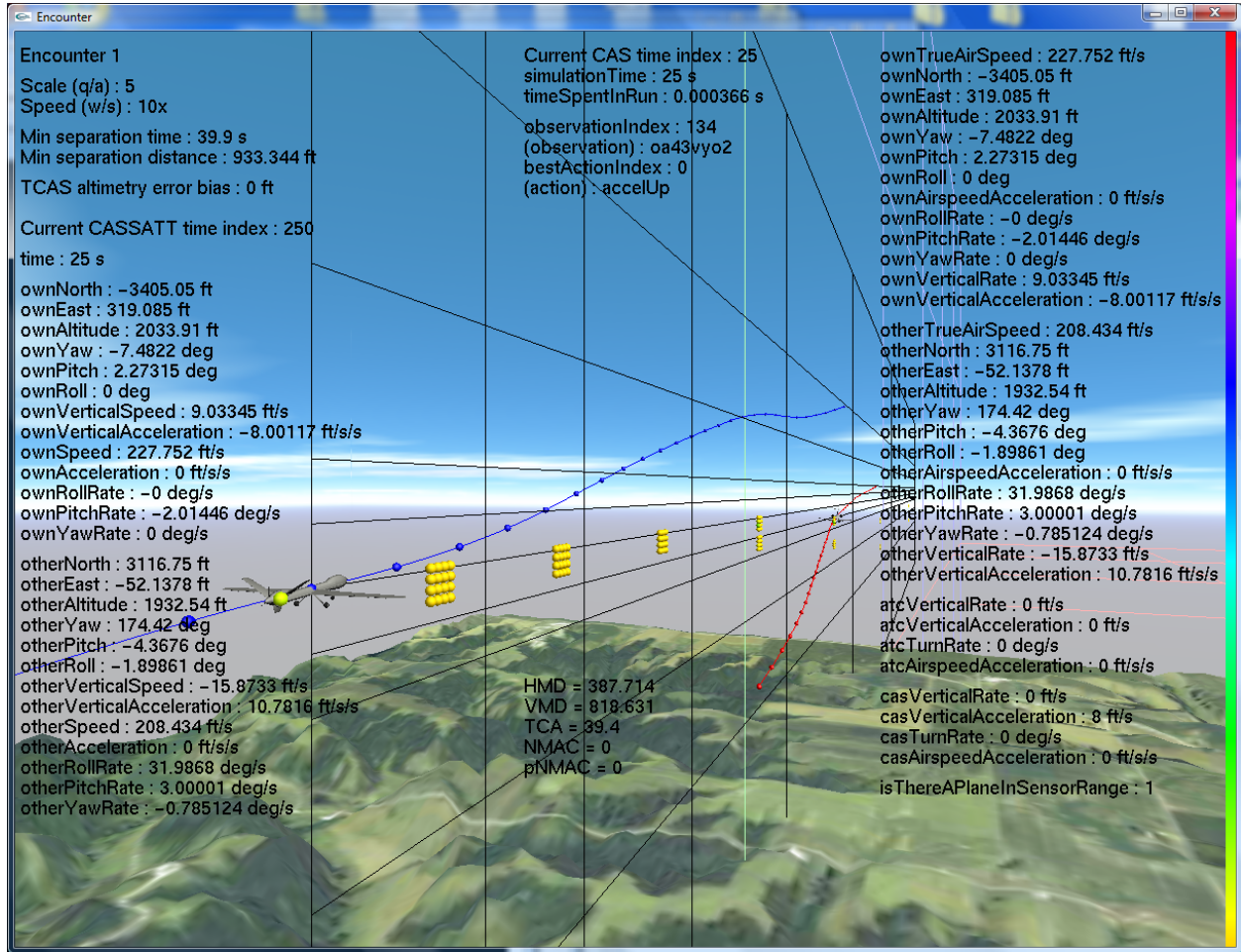


Figure C-1. A screenshot of the Encounter Analyzer showing flight trajectory and debugging information.

It is difficult to debug such a system using conventional software debugging techniques. An important part of debugging a POMDP during a simulation run is understanding the current state, which can be challenging using conventional tools because our POMDPs have over 2000 states. To aid in debugging, we created a visualization tool called “Encounter Analyzer” (Figure C-1).

The data to be visualized is acquired from two sources:

- At the end of each encounter, CASSATT outputs a special data structure that contains all CASSATT generated data. We prepared a Matlab script that reads in this special data structure, and outputs a text file that Analyzer can parse. The size of this text file is usually around 140–150 KB.
- If data logging is turned on, the controller dumps all its inputs, outputs and internal data to a text file—except for its internal belief-state (since it is large and can be readily recomputed). The size of this text file is usually around 30–40 KB.

Each entry in both of the text files is labeled with the simulation time it belongs to, so Analyzer can synchronize the two sources of data. Analyzer creates its own internal belief-state, and uses the recorded actions and observations to update it, so this belief-state is also synchronized with the actual belief-state used (but not recorded) by the controller.

Analyzer is built on top of the OpenGL library and it has both 3-dimensional graphing and head-up display (HUD) capabilities. The aim of Analyzer is to use graphical visualization (drawn to scale) as much as possible, and use HUD to display the rest of the data that cannot be drawn graphically.

The graphical visualizations can be grouped into 3 categories:

1. Visualizations of some of the gathered data
 - Own aircraft (a Predator B model is used)
 - Intruder aircraft (an Embraer Bombardier CRJ-200 model is used)
 - Trajectory of own aircraft for the entire encounter
 - Trajectory of intruder for the entire encounter
 - The active region for the sensor
 - Sensor reading
2. Other visual enhancements
 - A 3-D coordinate system
 - A tile of terrain
 - A sky-box
 - A bounding box that contains both trajectories
 - Projections of trajectories on the sides of the bounding box
 - A separate window that shows the same (synchronized) visual description as perceived by the pilot of our aircraft (there is a small offset that allows our aircraft to be in the view, too)
 - Another separate window that shows the same (synchronized) visual description as perceived by the intruder (there is a small offset that allows the intruder to be in the view, too)
3. Visualizations of data structures and computed variables
 - State space (relative coordinate system)
 - Action space
 - Observation space
 - Belief-state (states are colored, different colors indicate different probabilities, a “color legend” is also displayed on the right side of the main window)

- Computed observation
- Computed action

The textual (HUD) visualizations include the following:

- Current simulation time
- Aircraft state vectors for both aircraft (recorded from CASSATT special data structure)
- Aircraft state vectors for both aircraft (recorded by controller)
- ATC command
- Computed observation
- Computed action
- Computed aircraft control command
- Quantitative measure produced by CASSATT

Analyzer has a fourth window that displays a list of names of all of the graphical and textual visualizations described above. The user can click any item to toggle its visibility.

Analyzer allows the user to play (forward or backward) a recorded encounter step by step. At any step, one may examine the values of different variables, check the inputs and outputs of the controller, and move the camera around in the scene. Additionally, one may visualize the positions of the aircraft, their orientation, their sensor readings, and their belief states.

REFERENCES

- [1] 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).
- [2] T. Smith and R.G. Simmons, “Point-based POMDP algorithms: Improved analysis and implementation,” in *Proc. Int. Conf. on Uncertainty in Artificial Intelligence (UAI)* (2005).
- [3] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, “Online planning algorithms for POMDPs,” *Journal of Artificial Intelligence Research* 32, 663–704 (2008).
- [4] MITRE, “System safety study of minimum TCAS II,” MITRE, Technical Rep. MTR-83W241 (1983).
- [5] A. Drumm, “Lincoln Laboratory evaluation of TCAS II Logic Version 6.04a,” MIT Lincoln Laboratory, Project Report ATC-240 (1996).
- [6] M.P. McLaughlin, “Safety study of the Traffic Alert and Collision Avoidance System (TCAS II),” MITRE Corporation, Technical Rep. MTR 97W32 (1997).
- [7] B. Chludzinski, “Lincoln Laboratory evaluation of TCAS II logic version 7,” MIT Lincoln Laboratory, Project Report ATC-268 (1999).
- [8] RTCA, “Safety analysis of proposed change to TCAS RA reversal logic, DO-298,” RTCA, Inc., Washington, D.C. (2005).
- [9] T. Billingsley, *Safety Analysis of TCAS on Global Hawk using Airspace Encounter Models*, Master’s thesis, Massachusetts Institute of Technology (2006).
- [10] M.J. Kochenderfer, L.P. Espindle, J.K. Kuchar, and J.D. Griffith, “Correlated encounter model for cooperative aircraft in the national airspace system,” MIT Lincoln Laboratory, Project Report ATC-344 (2008).
- [11] J.K. Kuchar, “Methodology for alerting-system performance evaluation,” *Journal of Guidance, Control, and Dynamics* 19(2), 438–444 (1996).
- [12] L.F. Winder and J.K. Kuchar, “Evaluation of collision avoidance maneuvers for parallel approach,” *Journal of Guidance, Control, and Dynamics* 22(6), 801–807 (1999).

This page intentionally left blank.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE 22 September 2009		2. REPORT TYPE Project Report		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE Unmanned Aircraft Collision Avoidance Using Partially Observable Markov Decision Processes				5a. CONTRACT NUMBER FA8721-05-C-0002	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER 1	
6. AUTHOR(S) Selim Temizer, Mykel J. Kochenderfer, Leslie P. Kaelbling, Tomas Lozano-Perez, and James K. Kuchar				5d. PROJECT NUMBER	
				5e. TASK NUMBER 9433	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) MIT Lincoln Laboratory 244 Wood Street Lexington, MA 02420-9108				8. PERFORMING ORGANIZATION REPORT NUMBER ATC-356	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force ESC/PKE 5 Eglin Street Hanscom AFB, 01731				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) ESC-TR-2007-073	
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Before unmanned aircraft can fly safely in civil airspace, robust airborne collision avoidance systems must be developed. Instead of hand-carrying a collision avoidance algorithm for every combination of sensor and aircraft configuration, this project investigates the automatic generation of collision avoidance logic given models of aircraft dynamics, sensor performance, and intruder behavior. By formulating the problem of collision avoidance as a partially observable Markov decision process (POMDP), a generic POMDP solver can be used to generate avoidance strategies that optimize a cost function that balances flight-plan deviation with collision. Experimental results demonstrate the suitability of such an approach using three different sensor modalities and two aircraft performance models.					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Same as report	18. NUMBER OF PAGES 62	19a. NAME OF RESPONSIBLE PERSON
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (include area code)

