Robotics: Science and Systems 2011
Los Angeles, CA, USA, June 27-30, 2011

Unmanned Aircraft Collision Avoidance using Continuous-State POMDPs

Haoyu Bai* David Hsu* Mykel J. Kochenderfer† Wee Sun Lee*

*Department of Computer Science
National University of Singapore
Singapore, 117417, Singapore

†Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, MA 02420, USA

***Abstract*—An effective collision avoidance system for unmanned aircraft will enable them to fly in civil airspace and greatly expand their applications. One promising approach is to model aircraft collision avoidance as a partially observable Markov decision process (POMDP) and automatically generate the threat resolution logic for the collision avoidance system by solving the POMDP model. However, existing discrete-state POMDP algorithms cannot cope with the high-dimensional state space in collision avoidance POMDPs. Using a recently developed algorithm called Monte Carlo Value Iteration (MCVI), we constructed several continuous-state POMDP models and solved them directly, without discretizing the state space. Simulation results show that our 3-D continuous-state models reduce the collision risk by up to 70 times, compared with earlier 2-D discrete-state POMDP models. The success demonstrates both the benefits of continuous-state POMDP models for collision avoidance systems and the latest algorithmic progress in solving these complex models.**

I. Introduction

Unmanned aircraft have great potential for military, scientific, and commercial applications, but currently they cannot fly in civil airspace without special authorization. One primary concern is that unmanned aircraft do not yet have the capability to *sense and avoid* other aircraft effectively. An automated airborne collision avoidance system that meets the strict safety requirements of civil aviation authorities will greatly expand the applications of unmanned aircraft.

A key component of a collision avoidance system is the *threat resolution logic*, which relies on noisy sensor readings to detect other aircraft and must act under tight time limits to help bring the aircraft to safety. The safety-critical and time-critical nature of collision avoidance systems makes designing effective logic a significant challenge.

The Traffic Alert and Collision Avoidance System (TCAS) [13] is the most widely deployed collision avoidance system today and is mandated for all large commercial passenger and cargo aircraft worldwide. TCAS relies on dedicated transponders to sense nearby aircraft and uses complex, handcrafted threat resolution logic. TCAS is unsuitable for many types of unmanned aircraft due to constraints such as sensor cost and payload. Unmanned aircraft tend to use cheaper, lighter, but noisier sensors.

Unmanned aircraft collision avoidance is a challenging instance of planning under uncertainty. Uncertainties arise from sensor noise as well as the unknown flight dynamics and intentions of other aircraft. The partially observable Markov decision process (POMDP) is a principled and general framework for modeling and planning under uncertainty [10, 22]. The POMDP approach to collision avoidance system development involves building a model that specifies the goal and the operating environment of the system. The threat resolution logic is then generated automatically by solving the model.

This automated approach has several advantages over the traditional approach of manually designing the logic. It is nearly impossible for a human designer to anticipate all possible aircraft encounter situations, but the POMDP approach systematically accounts for all of them and their likelihood when solving the model. Once a successful POMDP model has been developed, it can be adapted relatively easily for different aircraft dynamics and sensor characteristics. The logic is then regenerated. It would be expensive and time consuming to handcraft a new collision avoidance system for every combination of aircraft platform and sensor configuration and to verify the safety of the system.

POMDPs, however, are often avoided in robotics because of high computational complexity [15]. In recent years, point-based algorithms have drastically improved the speed of POMDP planning by computing approximate solutions [14, 16, 23, 24]. The fastest offline POMDP algorithms today, such as HSVI [23] and SARSOP [14], can solve moderately complex POMDPs with nearly 100,000 states in reasonable time. However, these algorithms typically assume a discrete state space, while the natural state space of a robotic system is often continuous. Discretizing a continuous state space with a regular grid is not always practical. In a collision avoidance system, the state must contain the 3-D positions and velocities of at least two aircraft. Even a coarse discretization of the state space may result in hundreds of millions of states, which are beyond the reach of the fastest discrete-state POMDP algorithms today.

This work uses Monte Carlo Value Iteration (MCVI) [2], a recent point-based algorithm, to compute the threat resolution

The work of D. Hsu and W.S. Lee is supported in part by MoE AcRF grant 2010-T2-2-071 and MDA GAMBIT grant R-252-000-398-490. The portions of this work done by M.J. Kochenderfer are sponsored by the Department of Homeland Security under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

logic from a *continuous-state* POMDP model. MCVI solves the POMDP and outputs a *policy graph*. A policy graph is a directed graph in which each node represents an aircraft maneuver command and each edge represents a sensor observation. In the collision avoidance system, the policy graph can be implemented as a finite state controller that issues a command in response to a sensor observation. MCVI is well suited for computing the logic for several reasons:

- MCVI operates directly on continuous-state POMDP models and avoids inefficient a priori discretization of the state space.
- The resulting finite state controllers can be executed efficiently—using table lookup—to meet the strict time limits of collision avoidance systems.
- The graphical representation of policy graphs facilitates manual inspection and modification when necessary. They can also be validated using model-checking and simulation tools.

We constructed several collision avoidance POMDPs in both 2-D and 3-D spaces and with different sensors. We applied MCVI to each model to generate the logic and evaluated it in the CASSATT simulator [12], which is designed for testing collision avoidance systems. Simulation results show that our 3-D continuous-state models reduce the collision risk by up to 70 times, compared with 2-D discrete-state POMDP models in earlier work [25] and TCAS. The success demonstrates both the benefits of continuous-state POMDP models for collision avoidance systems and the latest algorithmic progress in solving these complex models.

In the following, Section II reviews related work. Section III presents our POMDP models. Section IV summarizes the MCVI algorithm for continuous-state POMDPs. Section V presents the simulation results. Section VI investigates why MCVI works so well on collision avoidance POMDPs. Section VII ends with some remarks on future directions.

II. Related Work

A. Aircraft Collision Avoidance Systems

Interest in collision avoidance systems dates back to the 1950s, when a mid-air collision of two aircraft occurred in the United States [13]. After decades of development, TCAS is currently the most widely used aircraft collision avoidance system. Various approaches have been proposed for collision avoidance, including geometric methods [6], potential field methods [7], and rapidly expanding random trees [20]. Compared with these alternatives, a major advantage of the POMDP approach is that it systematically accounts for uncertainties in the system.

Both offline and online algorithms have been proposed for collision avoidance POMDPs [25, 27]. As mentioned earlier, offline discrete-state POMDP algorithms have difficulty in scaling up to high-dimensional state spaces. Online algorithms usually handle high-dimensional state spaces well, but require significant online computation time [27], making it difficult to meet the real-time requirements of collision avoidance.

B. POMDP Algorithms

In a POMDP, the system state is not known exactly due to control and sensing errors. Instead, it is modeled as a *belief*, which is a probability distribution over possible system states. An algorithm solves a POMDP by computing an *optimal policy*, which specifies the best action for each belief.

In recent years, point-based algorithms made dramatic progress in computing approximate solutions to large discrete-state POMDPs, but progress on continuous-state POMDPs has been limited. A major difficulty is the belief and the policy representation, when high-dimensional continuous state spaces are involved. It is well known that representing and reasoning about high-dimensional, continuous probability distributions are difficult. One common approach is to restrict beliefs to a particular parametric form, *e.g.*, a Gaussian [4, 18] or a Gaussian mixture [5, 17]. To relax this assumption, other algorithms use particles to represent beliefs [17, 26], but it remains difficult to represent policies effectively.

MCVI is a point-based continuous-state POMDP algorithm that has shown good performance on several robot motion planning tasks, including navigation, grasping, and exploration [2]. MCVI uses particles to represents beliefs, thus making no assumption on their parametric form. It represents the policy as a policy graph [8], whose benefits for collision avoidance system modeling have already been outlined in Section I.

MCVI is an offline algorithm. Online search [9, 19, 21, 27] is complementary and can be used to further improve the performance of policies computed offline.

III. Collision Avoidance Models

A. Overview

A POMDP models a system taking a sequence of actions under uncertainty to maximize its total reward. Formally, a POMDP is a tuple $(S, A, O, T, Z, R, \gamma)$, where S, A, and O denote the system's state space, action space, and observation space, respectively. At each time step, the system takes an action $a \in A$ to move from a state $s \in S$ to $s' \in S$ and then receives an observation $o \in O$. A conditional probability function $T(s, a, s') = p(s'|s, a)$ models the state-transition dynamics. It specifies the probability distribution of the system state if the system takes action a in state s. Similarly, the conditional probability function $Z(s', a, o) = p(o|s', a)$ models the sensor observations. In a POMDP, the belief over system states is typically represented as a probability distribution $b(s)$ over S. The functions T and Z determine how to update $b(s)$ based on system dynamics and observations, respectively

At each time step, the system receives a real-valued reward $R(s, a)$ if it takes action $a \in A$ in state $s \in S$. The goal of the system is to choose a sequence of actions that maximizes the expected total reward $\mathrm{E}\big(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\big)$, where s_t and a_t denote the agent's state and action at time t, respectively, and $\gamma \in (0, 1)$ is a discount factor, which reflects that immediate rewards are preferred over future rewards.

Our models focus on situations involving two aircraft, the *own aircraft* and the *intruder aircraft*. The own aircraft has

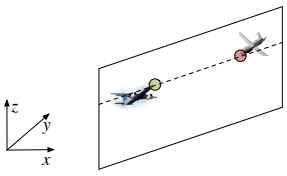


Fig. 1. An encounter model for two aircraft.

no prior information on the intruder aircraft's flight path, but has sensors onboard. Given noisy sensor input, the collision avoidance system maneuvers the own aircraft and tries to prevent Near Mid-Air Collision (NMAC) by keeping a safe separation distance between the two aircraft.

In our models, the state space is continuous, but the action and the observation spaces are discretized, due to the limitation of the MCVI algorithm. The details on state-transition, observation, and reward modeling are described below.

B. Flight Dynamics Modeling

We use a simplified flight dynamics model that treats an aircraft as a point mass in 3-D space (Fig. 1). Let (x, y, z) be the position of the aircraft with respect to the earth coordinate system, where the positive x-direction points east, the positive y-direction points north, and z is the altitude. Let θ be the aircraft's heading angle with respect to east. Let u and v be the aircraft's horizontal and vertical speed, respectively. The flight state of the aircraft is specified as (x, y, z, θ, u, v).

The aircraft control consists of the vertical acceleration $a \in \{-a_{\mathrm{m}}, 0, a_{\mathrm{m}}\}$ and the turn rate $\omega \in \{-\omega_{\mathrm{m}}, 0, \omega_{\mathrm{m}}\}$, where a_{m} and ω_{m} are the maximum control input values. Although aircraft control inputs are in fact continuous, restricting to the extreme values is reasonable for emergency maneuvers. This discretization helps improve computational efficiency during the POMDP policy computation. The horizontal speed u is constant and cannot be controlled. This is a reasonable simplification, as aircraft typically fly at high horizontal speed, which cannot be changed quickly under emergency conditions.

Given (a, ω), the new state of the aircraft after a small time duration Δt is given by

$$\begin{aligned} x_{t+1} &= x_t + u\Delta t \cos\theta, & \theta_{t+1} &= \theta_t + \omega\Delta t, \\ y_{t+1} &= y_t + u\Delta t \sin\theta, & u_{t+1} &= u_t, \\ z_{t+1} &= z_t + v\Delta t, & v_{t+1} &= v_t + a\Delta t. \end{aligned} \tag{1}$$

C. Encounter Modeling

An encounter model specifies the state-transition dynamics of the own aircraft and the intruder aircraft during an encounter. In our model, a state consists of the two aircraft's flight states. There are 9 discrete actions $\{-a_{\mathrm{m}}, 0, a_{\mathrm{m}}\} \times \{-\omega_{\mathrm{m}}, 0, \omega_{\mathrm{m}}\}$, which maneuver the own aircraft by specifying the control inputs for the vertical acceleration a and the turn rate ω. Because the intruder aircraft's control inputs a and ω are unknown, they are modeled as uniform random variables. Each aircraft's state-transition dynamics is then given by (1).

Although our encounter model has 12 state variables, the state space is effectively 10-dimensional since each aircraft's horizontal speed is constant. To cope with the high-dimensional state space, earlier work using a discrete-state POMDP approach had to adopt a lower-dimensional model and a coarse discretization [25]. First, the aircraft are not allowed to *turn*, so that they move entirely within a 2-D plane (Fig. 1) instead of the 3-D space. Next, the intruder aircraft's flight state is specified in the own aircraft's local coordinate system, in order to further reduce state-space dimensionality. Finally, the state space is discretized coarsely to fit within the capability of discrete-state POMDP algorithms.

Discretizing continuous state-transition dynamics introduces modeling errors that are often difficult to quantify. To find effective discretization, domain knowledge and many trials are needed to tune the granularity of discretization for each state variable. Our continuous-state POMDP models avoid these difficulties entirely and enable designers to specify aircraft flight dynamics in a much more natural and realistic manner.

TABLE I
SENSOR PARAMETERS.

	EO/IR	Radar
Range limit	5 NM[1]	5 NM
Azimuth limit	±110 deg	±110 deg
Elevation limit	±15 deg	±15 deg
Range error standard deviation	n/a	50 ft
Bearing error standard deviation	0.5 deg	1 deg
Elevation error standard deviation	0.5 deg	1 deg
False positive probability	0.01	0.01
False negative probability	0.01	0.01

[1]Nautical mile.

D. Sensor Modeling

We consider two sensor modalities likely suitable for unmanned aircraft: electro-optical/infrared (EO/IR) and passive radar.

EO/IR measures the intruder aircraft's elevation and bearing in the own aircraft's local coordinate system. The sensor's limited field of view (FoV), especially in the vertical direction (see Table I), makes aircraft control and sensing challenging. For computational efficiency, our observation model discretizes elevation and bearing into four equally spaced values each. When the sensor detects the intruder aircraft, it produces a pair of elevation-bearing values. Including the observation NO-DETECTION, there are 17 possible observations. The sensor may produce false positive or false negative detection, as well as error in elevation-bearing values. Our observation model accounts for all sensing errors. The main model parameters are shown in Table I. Implementation details of the model are the same as those in [25].

Radar is not as accurate as EO/IR in measuring elevation and bearing, but it measures the range to the intruder aircraft. The range is discretized into 3 values. The observation model for radar has 49 observations in total. Other aspects of the radar model are similar to those for EO/IR.

E. Reward Modeling

The primary goal of collision avoidance systems is to minimize the risk of NMAC. By definition, NMAC occurs when two aircraft are within a horizontal distance of 500 ft and

a vertical distance of 100 ft. Our reward model assigns a large penalty of $-10,000$ for NMAC. To discourage unnecessary maneuvers, our model assigns a penalty of -0.1 when the own aircraft has non-zero vertical velocity or turn rate.

IV. Monte Carlo Value Iteration (MCVI)

Now we briefly describe the algorithm used to solve collision avoidance POMDPs. The details are available in [2].

A. Policies and Value Functions

A POMDP algorithm computes an *optimal policy* π^* that maximizes a system's expected total reward. In a POMDP, the system state is partially observable and modeled as a belief, *i.e.*, a probability distribution over the state space S. Let $\mathcal{B}$ denote the space of all beliefs. A policy is a mapping $\pi\colon \mathcal{B} \to A$, which specifies an action $a \in A$ for each belief $b \in \mathcal{B}$.

A policy π induces a *value function* $V_\pi\colon \mathcal{B} \to \mathbb{R}$. The *value* of $b \in \mathcal{B}$ with respect to π is the system's expected total reward of executing π with initial belief b:

$$V_\pi(b) = \mathrm{E}\Big(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \;\Big|\; \pi, b\Big). \tag{2}$$

If the action space A and the observation space O of a POMDP are discrete, then the optimal value function V^* can be approximated arbitrarily closely by a piecewise-linear, convex function [17]:

$$V(b) = \max_{\alpha \in \Gamma} \int_{s \in S} \alpha(s) b(s)\, ds, \tag{3}$$

where each $\alpha \in \Gamma$ is a function over S and commonly called an *α-function*. If the state space S is also discrete, we can represent beliefs and α-functions as vectors and replace the integral in (3) by a sum. For each fixed α, $h(b) = \sum_{s \in S} \alpha(s) b(s)$ defines a hyperplane over $\mathcal{B}$, and V can be represented as a finite set of hyperplanes. Most of the fastest discrete-state POMDP algorithms [14, 16, 23, 24] represent a policy by its value function and exploit the vector representation for efficient computation. Unfortunately, when S is continuous and high-dimensional, a vector representation is no longer possible.

An alternative policy representation is a *policy graph* G, which is a labeled directed graph. Each node of G is labeled with an action a. Each edge of G is labeled with an observation o. To execute a policy π_G represented this way, we use a finite state controller whose states are the nodes of G. The controller starts in a node v of G, and the system, with initial belief b, performs the associated action a_v. If the system then receives an observation o, the controller transitions from v to a new node v' by following the edge (v, v') with label o. The process then repeats. The finite state controller does not maintain a belief on the system state explicitly. It encodes the belief implicitly in the controller state based on the initial belief b and the sequence of observations received.

For each node v of G, we may define an α-function α_v. Let $\pi_{G,v}$ denote a policy represented by G, when the controller always starts in node v of G. The value $\alpha_v(s)$ is the expected total reward of executing $\pi_{G,v}$ with initial state s:

$$\alpha_v(s) = \mathrm{E}\Big(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\Big) = R(s, a_v) + \mathrm{E}\Big(\sum_{t=1}^{\infty} \gamma^t R(s_t, a_t)\Big). \tag{4}$$

Putting (4) together with (2) and (3), we define the value of b with respect to π_G as

$$V_G(b) = \max_{v \in G} \int_{s \in S} \alpha_v(s) b(s) ds. \tag{5}$$

So V_G is completely determined by the α-functions associated with the nodes of G.

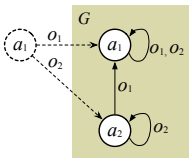


Fig. 2. MC-Backup of a policy graph G. The dashed lines indicate the new node and edges.

B. MC-Backup

The optimal value function V^* can be computed by the *value iteration* (VI) algorithm, which is based on the idea of dynamic programming. An iteration of VI is called a *backup*. The backup operator H constructs a new value function V_{t+1} from the current value function V_t:

$$V_{t+1}(b) = HV_t(b) = \max_{a \in A} \Big\{ R(b, a) + \gamma \sum_{o \in O} p(o|b, a) V_t(b') \Big\}, \tag{6}$$

In the equation above, $R(b, a) = \int_{s \in S} R(s, a) b(s) ds$ is the expected immediate reward, and b' is the belief on the next system state, after the system takes action a and receives observation o:

$$b'(s') = \tau(b, a, o) = \eta Z(s', a, o) \int_{s \in S} T(s, a, s') b(s)\, ds,$$

where η is a normalizing constant that ensures $\sum_{s' \in S} b(s') = 1$. At every $b \in \mathcal{B}$, the backup operator H looks ahead one step and chooses the action that maximizes the sum of the expected immediate reward and the expected total reward at the next belief. Under fairly general conditions, V_t converges to the unique optimal value function V^*.

VI is intended for computing value functions, but, interestingly, it can be carried out on policy graphs as well. Let V_G be the value function for a policy graph G. Substituting (5) into (6), we get

$$HV_G(b) = \max_{a \in A} \Big\{ \int_{s \in S} R(s, a) b(s) ds + \sum_{o \in O} p(o|b, a) \max_{v \in G} \int_{s \in S} \alpha_v(s) b'(s) ds \Big\}. \tag{7}$$

Let us now evaluate (7) at a particular point $b \in \mathcal{B}$ and construct the resulting new policy graph G', which contains a new node u and a new edge from u for each $o \in O$ (Fig. 2). Since we do not maintain V_G explicitly as a set of α-functions,

Algorithm 1 MC-Backup of a policy graph G at a belief $b \in \mathcal{B}$ with N samples.

MC-BACKUP(G, b, N)

1: For each action $a \in A$, $R_a \leftarrow 0$.
2: For each action $a \in A$, each observation $o \in O$, and each node $v \in G$, $V_{a,o,v} \leftarrow 0$.
3: **for** each action $a \in A$ **do**
4: **for** $i = 1$ to N **do**
5: Sample a state s_i with probability $b(s_i)$.
6: Simulate taking action a in state s_i. Generate the new state s'_i by sampling from the distribution $T(s_i, a, s'_i)$. Generate the resulting observation o_i by sampling from the distribution $Z(s'_i, a, o_i)$.
7: $R_a \leftarrow R_a + R(s_i, a)$.
8: **for** each node $v \in G$ **do**
9: Simulate the policy represented by G, with initial controller state v and initial state s'_i, for L steps. Set the resulting total reward $V' = \sum_{t=0}^{L} \gamma^t R(s_t, a_t)$.
10: $V_{a,o_i,v} \leftarrow V_{a,o_i,v} + V'$.
11: **for** each observation $o \in O$ **do**
12: $V_{a,o} \leftarrow \max_{v \in G} V_{a,o,v}$.
13: $v_{a,o} \leftarrow \arg\max_{v \in G} V_{a,o,v}$.
14: $V_a \leftarrow (R_a + \gamma \sum_{o \in O} V_{a,o})/N$.
15: $V^* \leftarrow \max_{a \in A} V_a$.
16: $a^* \leftarrow \arg\max_{a \in A} V_a$.
17: Create a new policy graph G' by adding a new node u to G. Label u with a^*. For each $o \in O$, add the edge $(u, v_{a^*,o})$ and label it with o.
18: **return** G'.

it seems difficult to compute the integral $\int_{s \in S} \alpha_v(s) b'(s) ds$. However, the definition of α_v in (4) suggests computing the integral by Monte Carlo (MC) simulation: repeatedly sample a state s with probability $b'(s)$ and simulate the policy $\pi_{G,v}$. In fact, we can evaluate the entire right-hand side of (7) via sampling and MC simulation, and construct the new policy graph G' with value function $\hat{H}_b V_G$. We refer to this as the MC-backup of G at b (Algorithm 1).

Conceptually, Algorithm 1 considers all possible ways of generating G'. The new node u in G' has $|A|$ possible labels, and each outgoing edge from u has $|G|$ possible end nodes in G, where $|G|$ is the number of nodes in G (Fig. 2). Thus, there are $|A||G|^{|O|}$ candidates for G'. Each candidate graph G' defines a new policy $\pi_{G',u}$. We draw N samples to estimate the value of b for each candidate $\pi_{G',u}$. For each sample, we pick s from the state space S with probability $b(s)$. We run an MC simulation under $\pi_{G',u}$, starting from s, for L steps and calculate the total reward $\sum_{t=0}^{L} \gamma^t R(s_t, a_t)$. The simulation length L is chosen to be large enough so that the error due to the finite simulation steps is small after discounting. We then choose the candidate graph with the highest average simulation reward. This naive procedure requires an exponential number of samples.

Algorithm 1 computes the same result, but is more efficient: the three nested loops (lines 3–10) use only $N|A||G|$ samples. Furthermore, we can show that MC-backup approximates the standard VI backup (equation (7)) well, with error decreasing at the rate $\mathcal{O}(1/\sqrt{N})$:

Theorem 1 ([2]): Let $R_{\max}$ be the maximum magnitude of $R(s, a)$ over $s \in S$ and $a \in A$. Given a policy graph G and a point $b \in \mathcal{B}$, MC-BACKUP(G, b, N) produces an improved policy graph with value function $\hat{H}_b V_G$ such that for any $\tau \in (0, 1)$,

$$\left| HV_G(b) - \hat{H}_b V_G(b) \right| \leq \frac{2R_{\max}}{1-\gamma} \times \sqrt{\frac{2\big(|O| \ln |G| + \ln(2|A|) + \ln(1/\tau)\big)}{N}}, \quad (8)$$

with probability at least $1 - \tau$.

C. Algorithm

MCVI computes an approximation to an optimal policy by updating a policy graph G. To improve G, it samples beliefs incrementally and performs backups at selected sampled beliefs. MCVI shares the same basic structure with the SARSOP algorithm [14] for discrete POMDPs; however, it uses MC-backup and particle filtering to handle continuous state spaces. Below we give a summary of the algorithm and describe the specific upper and lower bounds used in the implementation of MCVI for our collision avoidance POMDPs.

Let $\mathcal{R} \subseteq \mathcal{B}$ be a subset of points reachable from a given initial belief $b_0 \in \mathcal{B}$ under arbitrary sequences of actions and observations. Following the recent point-based POMDP planning approach, MCVI samples a set of beliefs from this reachable space $\mathcal{R}$ rather than $\mathcal{B}$ for computational efficiency, as $\mathcal{R}$ is often much smaller than $\mathcal{B}$. The sampled beliefs form a tree $T_{\mathcal{R}}$. Each node of $T_{\mathcal{R}}$ represents a sampled belief $b \in \mathcal{R}$, and the root of $T_{\mathcal{R}}$ is the initial belief b_0. If b is a node of $T_{\mathcal{R}}$ and b' is a child of b in $T_{\mathcal{R}}$, then $b' = \tau(b, a, o)$ for some $a \in A$ and $o \in O$. By definition, the belief associated with every node in $T_{\mathcal{R}}$ lies in $\mathcal{R}$.

To sample new beliefs, MCVI updates $T_{\mathcal{R}}$ by performing a search in $\mathcal{R}$. At each node b of $T_{\mathcal{R}}$, it maintains both upper and lower bounds on $V^*(b)$. We start from the root of $T_{\mathcal{R}}$ and traverse a single path down until reaching a leaf of $T_{\mathcal{R}}$. At a node b along the path, we choose action a that leads to the child node with the highest upper bound and choose observation o that leads to the child node making the largest contribution to the gap between the upper and lower bounds at the root of $T_{\mathcal{R}}$. These heuristics are designed to bias sampling towards regions that will likely lead to improvement in the value function approximation. If b is a leaf node, then we use the same criteria to choose a belief b' among all beliefs reachable from b after a single action $a \in A$ and an observation $o \in O$. We compute $b' = \tau(b, a, o)$ using particle filtering and create a new node for b' in $T_{\mathcal{R}}$ as a child of b. The sampling path terminates when it reaches a sufficient depth to improve the bounds at the root of $T_{\mathcal{R}}$. We then go back up this path to the root and perform backup at each node along the way to update the policy graph G as well as to improve the upper and lower bound estimates. We repeat the sampling and backup procedures until the gap between the upper and lower bounds at the root of $T_{\mathcal{R}}$ is smaller than a pre-specified value.

The lower bound at a tree node b is computed from the policy graph G. Since G represents a valid policy, $V_G(b)$ is

TABLE II
PERFORMANCE COMPARISON OF THREAT RESOLUTION LOGIC.

Model	Algorithm	Sensor	Risk Ratio	v (ft/s)	a (ft/s^2)	$\lvert\pi\rvert$
3D continuous-state POMDP	MCVI	EO/IR	**0.000657**	27.85	1.89	59
2D continuous-state POMDP	MCVI		**0.017409**	31.00	1.22	362
2D discrete-state POMDP	SARSOP		0.035100	28.61	1.48	n/a
3D continuous-state POMDP	MCVI	Radar	**0.000892**	26.85	1.83	43
2D continuous-state POMDP	MCVI		**0.021739**	29.50	1.67	766
2D discrete-state POMDP	SARSOP		0.063370	23.63	1.26	n/a
TCAS II	TCAS		0.061220	5.09	0.35	n/a
Nominal	n/a		1.000000	4.25	0.17	n/a

a lower bound of $V^*(b)$. We initialize G with fixed-action policies, each performing a single fixed action $a \in A$. Since there are nine actions for our collision avoidance POMDPs, G initially contains nine nodes. To update the lower bound at b, we perform MC-backup on G at b. As a result, we obtain an updated policy graph G' and an MC estimate of the value at b with respect to G' as an improved lower bound.

In a collision avoidance POMDP, all reward values are negative. So we use the zero initial bound at a new belief node and update it with the standard point-based backup operator.

To summarize, the main technical innovation of MCVI is to use Monte Carlo sampling in conjunction with dynamic programming to compute an approximate POMDP solution. It captures the α-functions implicitly as a policy graph and retains their main benefits after paying a computational cost. The idea of approximate dynamic programming through sampling has also been used in policy search for Markov decision processes (MDPs) and POMDPs, but without exploiting the benefits of α-functions [1].

V. SIMULATION RESULTS

We constructed two collision avoidance POMDPs in 3-D space with EO/IR and radar sensors, respectively. We used MCVI to solve each model and evaluated the resulting policy in the CASSATT simulator [12]. CASSATT is a fast, parallelized simulator for testing threat resolution logic. It has been used to test TCAS for manned aircraft [3]. Our tests used the same 15,000-encounter data set from earlier work [25]. This data set was generated from a model derived from nine months of radar data in the United States airspace [11]. Each encounter lasts about 50 seconds, and the collision avoidance system generates a command every six seconds.

To compare with discrete-state POMDP models [25], we also constructed POMDPs in 2-D space by projecting the state-transition dynamics and observations of our 3-D models onto the vertical plane containing the two aircraft. The action space of the 2-D models contains the vertical acceleration v, but not the turn rate ω. In the CASSATT simulator, policies generated from 2-D models control only v, and the aircraft maintains its nominal turn rate.

Table II summarizes the simulation results. Column 1 and 2 list the model and the algorithm that produced the threat resolution logic. All POMDP models used the uniform initial belief and the same reward function described in Section III-E.

TABLE III
RISK RATIO VERSUS MANEUVER PENALTY.

R_{m}	Risk Ratio	v (ft/s)	a (ft/s^2)	ω (deg/s)
-0.1	0.000657	27.85	1.89	0.0269
-0.5	0.001297	25.14	2.42	0.0239
-1	0.002230	26.90	1.80	0.0264
-2	0.097492	23.92	2.09	0.0148
-5	0.247996	6.32	0.83	0.0065

For each continuous-state model, it took a maximum of 8 hours to compute a policy on a computer server with two 4-core 2.4 GHz CPUs. The results for the discrete-state models and TCAS are based on the reports in [25].

Column 4 of Table II lists the risk ratio, the main performance measure of collision avoidance systems. *Risk ratio* is the probability that an encounter leads to NMAC when using the system divided by that when not using the system. The last row of Table II shows the risk ratio for nominal flight, which maintains the current flight path without any collision avoidance maneuver. Column 5 and 6 list the mean magnitudes of vertical velocity v and acceleration a. It is desirable to have low velocity and acceleration without much sacrifice in the risk ratio. Column 6 lists the number of nodes in the policy graphs produced by MCVI.

Compared with 2-D discrete-state models, 2-D continuous-state models reduced the risk ratio by 2–3 times, even though both types of models have the same underlying flight dynamics, the same action space, and the same observation space.

Our 3-D continuous-state models reduced the risk ratio by 58 times with the EO/IR sensor and 70 times with the radar sensor. MCVI enabled us to model aircraft maneuver in the full 3-D space, allowing the risk ratio to be significantly reduced. It is impossible for discrete-state POMDPs to handle such high-dimensional state spaces.

All our models produced logic with lower risk ratio than the TCAS logic. Since TCAS is not optimized for automated unmanned aircraft control, a direct comparison is not fair, but this nevertheless indicates the potential of our approach.

We also examined how the level of maneuver affects the risk ratio by varying the maneuver penalty R_{m} in the 3-D continuous-state model with the EO/IR sensor. As expected, when the cost increases, the risk ratio increases, and the level of maneuver decreases, though the relationships are not exactly monotonic (Table III).

Fig. 3 shows a policy graph for our 3-D continuous-state

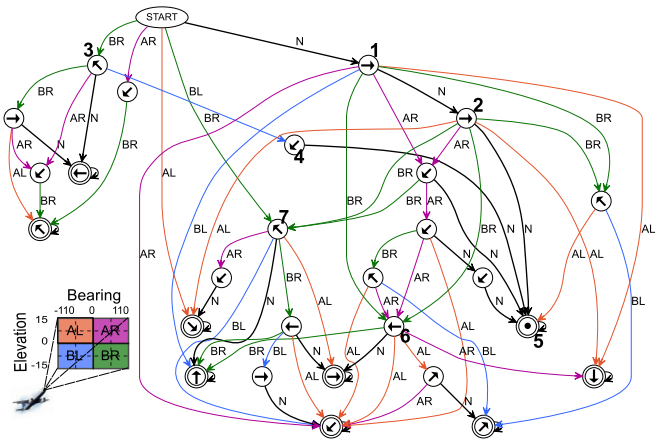


Fig. 3. A subgraph of the policy graph for the 3-D continuous-state POMDP model with EO/IR. Each node is labeled with START, an arrow, or a dot. START indicates the start node. An arrow indicates the directions of vertical acceleration and turn. A dot indicates zero acceleration and turn. Doubly-circled nodes represent initial policies. Each edge is color-coded and labeled with BL (below-left), BR (below-right), AL (above-left), AR (above-right), or N (no detection).

model with the EO/IR sensor. Only a subgraph can be shown in the limited space. It provides several interesting observations:

- The own aircraft executes various maneuvers for information gathering. When the observation is NO-DETECTION, the aircraft turns right for two steps (node 1 and 2), in order to overcome the sensor's limited FoV and scan a larger region for the intruder aircraft. The own aircraft may also ascend and then descend (node 3 and 4) based on the observations to localize the intruder aircraft better.
- After turning right for two steps (node 1 and 2), the aircraft switches to level flight (node 5) in response to further NO-DETECTION, rather than turn left to continue searching for the intruder aircraft. Several successive NO-DETECTION observations indicate that the space to the right is likely clear. Continuing the search is unlikely to be fruitful, whether the intruder aircraft is detected in the end or not. This shows the policy's ability to balance information exploration and exploitation, a key benefit of the POMDP approach.
- The policy graph contains several reused components, *e.g.*, the parts starting at node 6 and 7, respectively. They can be regarded as sub-policies.

VI. DISCUSSION

Computing an optimal policy to a POMDP is PSPACE-hard in the worst case [15]. Furthermore, under the assumption PSPACE $\neq \Sigma_2^p$, no polynomial-sized policy exists [15]. Surprisingly, MCVI found small and effective policies for collision avoidance POMDPs (see Table II). In this section, we try to find out the reasons.

Our main intuition is that once the intruder aircraft is detected, a simple policy is often sufficient to bring the own aircraft to safety. More precisely, consider the belief tree $T_{\mathcal{R}}$ for a POMDP with time horizon h. Conceptually, a planning algorithm, such as MCVI, chooses one action at each node

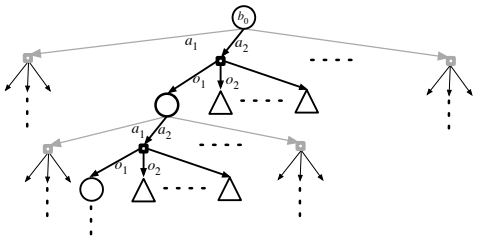


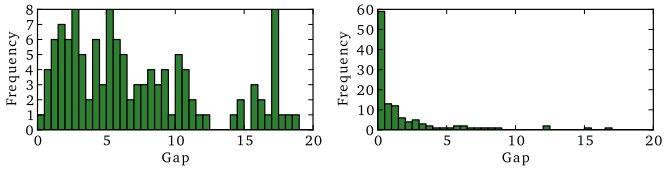
Fig. 4. The belief tree for an SD POMDP. The part in black indicates a policy that makes use of simple sub-policies, each represented as a "△".

of $T_{\mathcal{R}}$, thereby converting $T_{\mathcal{R}}$ into a policy tree T_π. Every node of T_π has at most $|O|$ branches, each corresponding to an observation $o \in O$, and the size of T_π is $\Theta(|O|^h)$ in the worst case. Assume now that all but one branch at each node of T_π admits a sub-policy of size at most M. This implies the existence of an optimal policy π^* with size at most $\Theta(|O|Mh)$, which is polynomial in h (Fig. 4). If the sub-policies are all polynomial-sized, then so is π^*. In this case, we say that the POMDP is *simple after detection* (SD).

It is reasonable to expect that in collision avoidance POMDPs, simple policies do exist for every branch of a belief tree node except for the one with NO-DETECTION. To verify that collision avoidance POMDPs are SD, we measured the gap between the upper and lower bounds of randomly sampled nodes from $T_{\mathcal{R}}$ for the 3-D continuous-state model with the EO/IR sensor, as small gap size indicates the existence of good sub-policies. Specifically, we randomly sampled NO-DETECTION paths of various lengths from the belief tree and performed a single backup operation at the last node along each path. We then measured the gaps between the upper and lower bounds of these nodes, using the same bounds as those in MCVI. We did the same with randomly sampled *first-detection* paths, which are paths having NO-DETECTION observations except for the last one. While the gap sizes for NO-DETECTION paths vary widely from small to large, the gap sizes for first-detection paths are all concentrated at the small end (Fig. 5). The sharp difference suggests that our collision avoidance POMDP models are approximately SD.

Various uncertainty planning tasks in robotics and beyond may possess the SD property. For example, once a robot finds a target, following the target around is often not difficult. Similarly, suppose that a robot has relatively accurate motion control. Once the robot is localized, navigation becomes easy.

A small policy does not imply that it can be computed efficiently, but the SD property indeed seems to bring substantial computational advantages to MCVI. The belief tree $T_{\mathcal{R}}$ for a POMDP with time horizon h has size $\Theta(|A|^h|O|^h)$ in the worst case. However, our measurements (Fig. 5) indicate that at each node of $T_{\mathcal{R}}$, the existence of simple sub-policies allows a single backup to substantially narrow the gap between the upper and lower bounds for all observation branches except for the one with NO-DETECTION. Thus MCVI can prune most branches of $T_{\mathcal{R}}$, and the size of $T_{\mathcal{R}}$ is only about $\Theta(|A|^h)$, which is exponentially smaller than the worst case. In our collision avoidance POMDPs, $|O|$ is 2–5 times as large as $|A|$,



(*a*) (*b*)

Fig. 5. Gap size distributions for the EO/IR model. (*a*) NO-DETECTION paths. (*b*) First-detection paths.

and the reduction in the size of $T_{\mathcal{R}}$ is significant. Furthermore, an encounter of two aircraft must be resolved within tight time limits. Thus, the time horizon h is relatively short. All these limit the size of $T_{\mathcal{R}}$ and contribute to the strong performance of MCVI.

VII. CONCLUSIONS

We modeled unmanned aircraft collision avoidance as continuous-state POMDPs and generated the threat resolution logic automatically by solving these models using Monte Carlo Value Iteration. Simulation results show that our 3-D continuous-state models reduce the collision risk by up to 70 times compared with previous 2-D discrete-state POMDP models. A key enabler of this significant progress is MCVI's ability in handling high-dimensional continuous state spaces. The combination of continuous-state POMDP modeling and the MCVI algorithm provides a powerful approach for collision avoidance and potentially a variety of other robotic tasks that require planning under uncertainty.

We are extending both our POMDP models and the MCVI algorithm in several directions. One important issue is to handle continuous observation space, which may further improve the performance of our models. It would also be interesting to apply our approach to collision avoidance systems for multiple aircraft and to explore interoperability with collision avoidance systems for manned aircraft.

Acknowledgments. We thank Tomás Lozano-Pérez, Leslie Kaelbling, and Selim Temizer from MIT for many insightful discussions.

REFERENCES

[1] J.A. Bagnell, S. Kakade, A. Ng, and J. Schneider. Policy search by dynamic programming. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16. 2003.

[2] H.Y. Bai, D. Hsu, W.S. Lee, and V.A. Ngo. Monte Carlo value iteration for continuous-state POMDPs. In D. Hsu et al., editors, *Algorithmic Foundations of Robotics IX—Proc. Int. Workshop on the Algorithmic Foundations of Robotics (WAFR)*. Springer, 2010.

[3] T. Billingsley. Safety analysis of TCAS on Global Hawk using airspace encounter models. Master's thesis, Massachusetts Institute of Technology, 2006.

[4] A. Brooks, A. Makarendo, S. Williams, and H. Durrant-Whyte. Parametric POMDPs for planning in continuous state spaces. *Robotics & Autonomous Systems*, 54(11):887–897, 2006.

[5] E. Brunskill, L. Kaelbling, T. Lozano-Perez, and N. Roy. Continuous-state POMDPs with hybrid dynamics. In *Proc. Int. Symp. on Artificial Intelligence & Mathematics*, 2008.

[6] G. Dowek, A. Geser, and C. Muñoz. Tactical conflict detection and resolution in a 3-D airspace. In *USA/Europe Air Traffic Management R&D Seminars*, 2001.

[7] V.N. Duong and K. Zeghal. Conflict resolution advisory for autonomous airborne separation in low-density airspace. In *Proc. IEEE Conf. on Decision & Control*, 1997.

[8] E.A. Hansen and S. Zilberstein. Heuristic search in cyclic AND/OR graphs. In *Proc. AAAI Conf. on Artificial Intelligence*, pages 412–418, 1998.

[9] R. He, E. Brunskill, and N. Roy. PUMA: Planning under uncertainty with macro-actions. In *Proc. AAAI Conf. on Artificial Intelligence*, 2010.

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

[11] M.J. Kochenderfer, L.P. Espindle, J.K. Kuchar, and J. D. Griffith. Correlated encounter model for cooperative aircraft in the national airspace system. Project Report ATC-344, Massachusetts Institute of Technology, Lincoln Laboratory, 2008.

[12] J.K. Kuchar. Safety analysis methodology for unmanned aerial vehicle (UAV) collision avoidance systems. In *USA/Europe Air Traffic Management R&D Seminars*, 2005.

[13] J.K. Kuchar and A.C. Drumm. The traffic alert and collision avoidance system. *Lincoln Laboratory Journal*, 16(2):277–296, 2007.

[14] H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems*, 2008.

[15] C. Papadimitriou and J.N. Tsisiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[16] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. Int. Jnt. Conf. on Artificial Intelligence*, pages 477–484, 2003.

[17] J.M. Porta, N. Vlassis, M.T.J. Spaan, and P. Poupart. Point-based value iteration for continuous POMDPs. *J. Machine Learning Research*, 7:2329–2367, 2006.

[18] S. Prentice and N. Roy. The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance. In *Proc. Int. Symp. on Robotics Research*, 2007.

[19] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa. Online planning algorithms for POMDPs. *J. Artificial Intelligence Research*, 32(1):663–704, 2008.

[20] J. Saunders, R. Beard, and J. Byrne. Vision-based reactive multiple obstacle avoidance for micro air vehicles. In *Proc. American Control Conf.*, 2009.

[21] D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[22] R.D. Smallwood and E.J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[23] T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proc. Uncertainty in Artificial Intelligence*, 2005.

[24] M.T.J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *J. Artificial Intelligence Research*, 24:195–220, 2005.

[25] S. Temizer, M.J. Kochenderfer, L.P. Kaelbling, T. Lozano-Pérez, and J.K. Kuchar. Collision avoidance for unmanned aircraft using Markov decision processes. In *Proc. AIAA Guidance, Navigation, and Control Conference*, 2010.

[26] S. Thrun. Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*. The MIT Press, 2000.

[27] T.B. Wolf and M.J. Kochenderfer. Aircraft collision avoidance using Monte Carlo real-time belief space search. *J. Intelligent & Robotic Systems*, 2011. To appear.