SHORT-TERM CONFLICT RESOLUTION FOR UNMANNED AIRCRAFT TRAFFIC MANAGEMENT

Hao Yi Ong and Mykel J. Kochenderfer, Stanford University, Stanford, CA

Abstract

Ensuring safety and providing timely conflict alerts to small unmanned aircraft is paramount to their integration into civil airspace. We propose several controllers to solve this stochastic short-term conflict avoidance problem, where a collection of unmanned aircraft is supervised by a traffic management system. The goal is to balance between aircraft safety and alert rates subject to environmental and aircraft uncertainty. The proposed controllers generate advisories for each aircraft to follow, and are based on decomposing a large Markov decision process and fusing their solutions. Specifically, we separate the problem into pairwise encounters that are solved exactly offline. In online operation, solutions to these encounters are combined to produce a locally optimal solution using an iterative search technique. As a result, the methods scale well and the global problem can be solved efficiently. We demonstrate the controllers by evaluating them against baseline algorithms in simulation.

1. Introduction

Many important applications of small unmanned aircraft, ranging from goods delivery to infrastructure surveillance, can be enabled by flexible access to civil airspace. To enable safe and efficient operations at low-altitude, NASA has been leading the exploration of concepts and prototypes for a largely automated UAS Traffic Management (UTM) system [1]. While there are many important components to the UTM, this paper focuses on automated conflict avoidance because it is critical to ensuring safety.

Designing a robust conflict avoidance system is challenging for a variety of reasons. The UTM is intended to handle many more aircraft in a more restrictive and crowded airspace than conventional air traffic management systems. There is uncertainty in the current state due to imperfect sensor measurements, and there is uncertainty in the future paths of the aircraft due to variability in pilot responses. Determining the appropriate trade-off between safety and alert rates is not straightforward. The system also must coordinate complementary advisories among many aircraft.

Past approaches to conflict avoidance problems include mixed-integer programming and sequential convex programming by Schouwenaars, Valenti, Feron, *et al.* [2], Mellinger, Kushleyev, and Kumar [3], and Augugliaro, Schoellig, and D'Andrea [4], respectively. These approaches tend to work well for small vehicle networks. Ong and Gerdes have explored convex optimization techniques to develop an efficient and scalable algorithm called proximal message passing [5]. Exploiting specific structures in short-term conflict resolution, Erzberger, Lauderdale, and Chu propose a deterministic algorithm to choose feasible trajectories out of a set for a pair of commercial aircraft [6], [7]. Airborne Collision Avoidance System X (ACAS X) uses a control policy derived from Markov decision process (MDP) solution techniques and has been shown to improve safety and operational efficiency over existing collision avoidance systems [8], [9]. The work by Owen, Williams, and Mezhirov extends ACAS X to UAS, but it does not coordinate horizontal maneuvers [10]. Kuchar and Yang offers a less recent but comprehensive survey of conflict avoidance methods in [11].

The contributions of this paper are threefold. First, we formulate the conflict resolution problem as a stochastic problem in the form of a large MDP and decompose it into computationally tractable subproblems. Second, we propose a locally optimal coordination scheme that is based on an iterative method that combines subproblem solutions [12]. Third, we develop several variants of the algorithm that permit solutions in real-time and demonstrate their effectiveness over two baseline methods.

2. Markov Decision Process

In an MDP, an agent chooses action a_t at time t after observing state s_t. The agent then receives

reward r_t, and the state evolves probabilistically based on the current state-action pair. The explicit assumption that the next state only depends probabilistically on the current state-action pair is referred to as the Markov assumption. An MDP can be defined by the tuple (S, A, T, R), where S and A are the sets of all possible states and actions, respectively, T is a probabilistic transition function, and R is a reward function. The probability of transitioning into state s' after taking action a from state s is denoted $T(s, a, s')$. The immediate reward received for taking action a from state s is denoted $R(s, a)$.

While solving MDPs is an effective method for determining actions for a single agent in stochastic environments, they can also be extended to cooperative multi-agent domains. Multi-agent MDPs (MMDPs) extend MDPs and allow for sequential decision-making in a cooperative multi-agent system, and is similar to the case where a centralized planner has access to the system state. Similar to an MDP, an MMDP can be defined by the tuple (S, A, T, R). The difference is that S and A are now the sets of all possible joint states and actions, respectively, and that T and R operate on elements from these sets.

To solve any MDP (or MMDP), we compute a policy $\pi^\star$ that, if followed, maximizes the expected discounted sum of immediate rewards from any given state. The optimal policy is related to the optimal state-action utility function $U^\star(s, a)$, which is the expected utility when starting in state s, taking action a, and then following actions dictated by $\pi^\star$. In the literature, this is also called the state-action value function. Mathematically, it obeys the Bellman recursion

$$U^\star(s, a) = R(s, a) + \sum_{s' \in S} T\left(s, a, s'\right) U^\star\left(s'\right), \quad (1)$$

where we define $U^\star(s) = \max_{a \in A} U^\star(s, a)$. The state-action utility function can be computed using a dynamic programming algorithm called value iteration. Beginning with an initial guess, this technique iteratively updates (1) until the estimate converges. The optimal policy, in turn, is given by

$$\pi^\star(s) = \underset{a \in A}{\operatorname{argmax}}\, U^\star(s, a). \quad (2)$$

Because the number of states scale exponentially with the number of state variables, exact solutions may be computationally intractable for problems with more than just a few state variables or agents [13]. This issue is referred to as the curse of dimensionality.

3. Short-term Conflict Avoidance

Our formulation focuses on horizontal or co-altitude conflict resolution. Here, we have n aircraft at risk of conflict between each other. Here, conflict is defined as the loss of minimum separation, and a pair of aircraft is considered at risk if it could experience a conflict within a time frame of up to two minutes. Loss of minimum separation, in turn, is defined as the situation when two aircraft come closer than some threshold distance.

A. Resolution Advisories

At each time step, the system issues a joint advisory out of a finite set of bank angles corresponding to each aircraft's action in the multi-threat scenario. The joint advisory can be written as $\phi = (\phi_1, \ldots, \phi_n)$. While the time period for a single transition may be defined according to system needs, we define it to be 5 s. This period corresponds to the decision frequency of our system. These bank angles are directly related to the turn radii of the aircraft through the Dubin kinematic equations. These joint advisories constitute the action set A^n. We define $A = \{-20°, -10°, 0°, 10°, 20°, \text{COC}\}$. Here, positive angles correspond to left banks and negative angles to right banks. COC is a special clear-of-conflict status that allows the aircraft to continue its trajectory as usual. Conversely, all other actions correspond to corrective advisories to be issued during conflict.

B. Dynamic Model

The system is described by the set of state variables of each aircraft. For the ith plane, of which there are n in total, the state is $s_i = (x_i, y_i, \psi_i, v_i)$, where x_i and y_i indicate the absolute geographical position, ψ_i indicates the absolute heading and v_i indicates the aircraft speed.

Each aircraft pilot responds immediately to the resolution advisory. The speed is held constant between decision stages. To account for a wide range of pilot responses including delayed reaction and aggressive over-reaction, we model the aircraft banking response as a Gaussian distribution. When the aircraft is in conflict, the bank angle distribution is centered on the resolution advisory with a standard deviation

of 4°. When the aircraft is clear of conflict, the aircraft is modeled to bank randomly based on a zero-mean Gaussian with a standard deviation of 10°.

The aircraft speed is modeled as a Gaussian distribution centered on the aircraft's reported speed with a standard deviation of 2 m/s. The variation in aircraft speed captures random effects like navigation errors, wind gusts, and other trajectory perturbations. This simplification is due to the observation that aircraft usually respond slowly to sudden speed change inputs for short durations. As mentioned, the formulation uses Dubin's kinematic model to compute state transitions. The model is described by

$$\dot{x} = v\cos\psi,\ \ \dot{y} = v\sin\psi,\ \ \dot{\psi} = \frac{g\tan\phi}{v}, \qquad (3)$$

where v is the speed and g is the gravitational acceleration. The relative simplicity of this dynamic model reduces the computation time required to solve the problem and the risk of overfitting a more complicated model. The latter is crucial, given the diversity of unmanned aircraft.

C. Reward Function

A conflict avoidance system must balance multiple objectives such as maximizing conflict avoidance and minimizing disruption. These objectives are captured in a reward function composed of a sum of different components. For safety, aircraft are not allowed to come within a separate minimum distance $r^{\min} = 500$ m. At any time t, the separation between two aircraft is denoted $r(t)$. The minimum separation between the two aircraft during each decision period T starting at time t is

$$r_t^{\text{sep}} = \min\{r(\tau) \mid \tau \in [t, t+T)\}.$$

The reward function penalizes any decision that brings any pair of aircraft to $r_t^{\text{sep}} < r^{\min}$. The function further penalizes the minimum separation of any two aircraft by an exponentially decaying graph centered at 0 m. Corrective resolution advisories are penalized over the clear-of-conflict advisory. To prevent aircraft from banking too much during resolution, nonzero corrective bank angles are further penalized. For notational convenience, the COC action corresponds to a 0° bank angle during reward computation. The reward function for a decision period starting at time t is written as

$$R_t(s,\phi) = -\lambda_1 I_{\text{sep}}(s,\phi) - \lambda_2 \exp\left(r_t^{\text{sep}}(s,\phi)/r^{\min}\right) \qquad (4)$$

$$-\lambda_3 \|\phi\|_2^2 - \lambda_4 1^T I_{\text{coc}}(\phi), \qquad (5)$$

where I_{sep} and I_{coc} are the indicator functions

$$I_{\text{sep}}(s,\phi) = \begin{cases} 1, & r_t^{\text{sep}}(s,\phi) < r^{\min} \\ 0, & \text{otherwise} \end{cases}$$

$$(I_{\text{coc}}(\phi))_i = \begin{cases} 1, & \phi_i \neq \text{COC} \\ 0, & \text{otherwise.} \end{cases}$$

For this algorithm, the weight parameters are $\lambda_1 = 1000$, $\lambda_2 = 10$, and $\lambda_3 = 0.02$. We balance between corrective resolution advisory alerts and safety (probability of conflict) by tuning λ_4, and we set $\lambda_4 = 10$ unless otherwise stated. While this reward function only captures basic preferences, a better function can be found through expert preference elicitation methods [14].

4. MDP Decomposition

As formulated in Section 3, computing the optimal solution is impractical. We therefore decompose the problem into pairwise encounter MDPs and fuse the state-action utilities together [9], [15].

A. Pairwise Conflict Avoidance

Instead of tackling the problem as a full MMDP, our approach first decomposes it into a set of $1/2n(n-1)$ pairwise encounters.

1) Resolution advisories: As in the original MMDP formulation, the system can issue one advisory out of a finite set of bank angles corresponding to turn radii for each aircraft at each time step. These joint advisories constitute the action set A, which follows the definition used in the MMDP.

2) Dynamic model: One may use the multi-agent formulation presented in Section 3 with only two aircraft. A better approach reduces the cardinality of the state space and thus computational complexity by replacing the absolute state variables of each aircraft with a single set of relative state variables. (The speeds, however, are still absolute with respect to the world.)

The pilot response and dynamic models of the aircraft is the same as in the original MMDP. Because we use relative states, however, appropriate modifications to the equations of motion are required

to take rotating reference frames into account. To compute the relative state transitions, (3) is used to update the absolute aircraft states from the initial configuration. We compute the intermediate states $x^{\text{abs}} = x_j - x_i$ and $y^{\text{abs}} = y_j - y_i$. The relative states $s^{\text{rel}} = \left(x^{\text{rel}}, y^{\text{rel}}, \psi^{\text{rel}}\right)$ for aircraft j with respect to i can then be computed as follows.

$$\begin{aligned}\psi^{\text{rel}} &= \psi_j - \psi_i \\ x^{\text{rel}} &= x^{\text{abs}} \cos\psi^{\text{rel}} + y^{\text{abs}} \sin\psi^{\text{rel}} \\ y^{\text{rel}} &= -x^{\text{abs}} \sin\psi^{\text{rel}} + y^{\text{abs}} \cos\psi^{\text{rel}}\end{aligned}$$

3) Reward function: The reward function for the pairwise encounter is equivalent to an MMDP formulation for two agents.

B. Offline Solution

The optimal policy specifies the target bank angle to reach and hold for the 5 s time step between decisions. States are mapped from the continuous to the discrete domain using multilinear interpolation, and transition probabilities are estimated using sigma-point sampling [16]. We use a set of weighted values for bank angle ϕ and speeds v_1 and v_2 for sigma-point sampling. For the transition between states, the system has a $1/3$ probability of following the nominal bank angle and speeds. The remaining $2/3$ weight is uniformly distributed across every other combination of $\left(\sigma_\phi, \sigma_{v_1}, \sigma_{v_2}\right)$, where $\sigma_\phi \in \{-4°, 0°, 4°\}$ for a corrective resolution advisory, $\sigma_\phi \in \{-10°, 0°, 10°\}$ for a clear-of-conflict status, and $\sigma_{v_{1|2}} \in \{-2\,\text{m/s}, 0\,\text{m/s}, 2\,\text{m/s}\}$. These values come from the Gaussian distributions over the bank angle response and aircraft speed in Section 3.

These techniques were effective in providing an approximate discrete alternative to an MDP with a continuous state space in [8], [9]. The state space for the relative variable formulation is discretized using a multidimensional grid. Table I shows the result of the discretization, which has approximately 870,000 discrete states. While the discretization can be made finer at the expense of additional computation and storage, this level of discretization was acceptable in numerical experiments.

Dynamic programming is used to compute the utility function $U^\star$. This technique is called value iteration, and solves the discretized problem optimally. The basic idea of this algorithm is to iteratively solve the Bellman recursion (1) from an initial guess. One

Table I: Discretization scheme

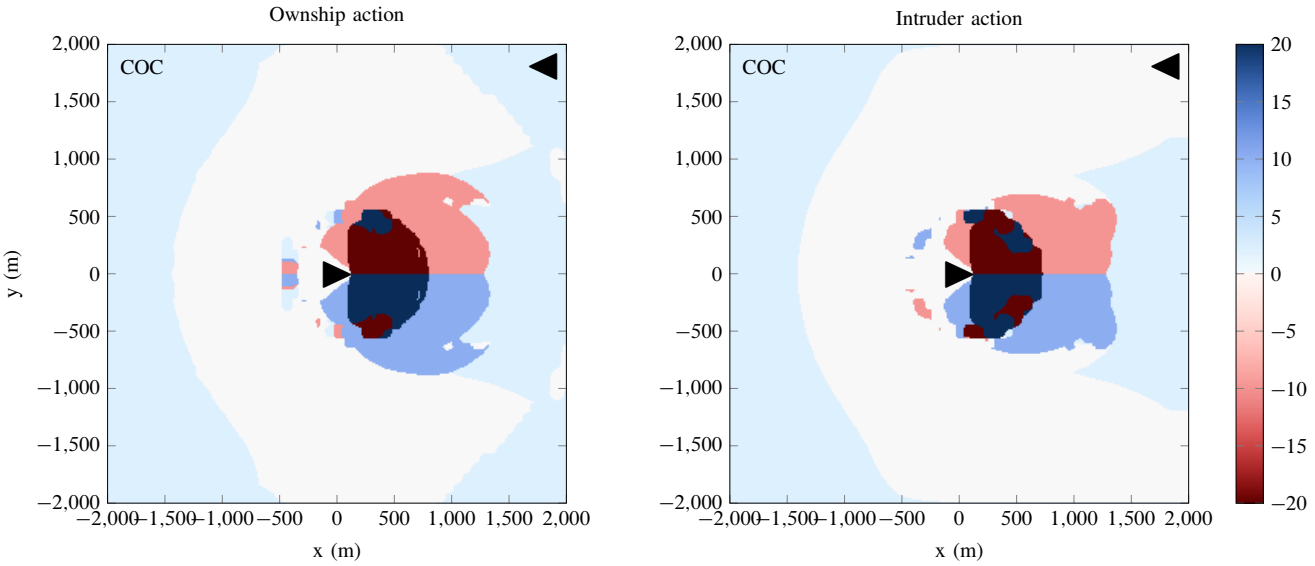
Variable	Minimum	Maximum	Number of values
x, y	$-3000\,\text{m}$	$3000\,\text{m}$	51
ψ	$0°$	$360°$	37
v_1, v_2	$10\,\text{m/s}$	$20\,\text{m/s}$	3

modification is that the original transition function is replaced by $\tilde{T}$, the approximate transition function resulting from sigma-point sampling and multilinear interpolation. In the case of partial state observability, a heuristic called QMDP can be used to produce a solution that performs well in many real-world scenarios where there are few information-gathering actions [17], [18].

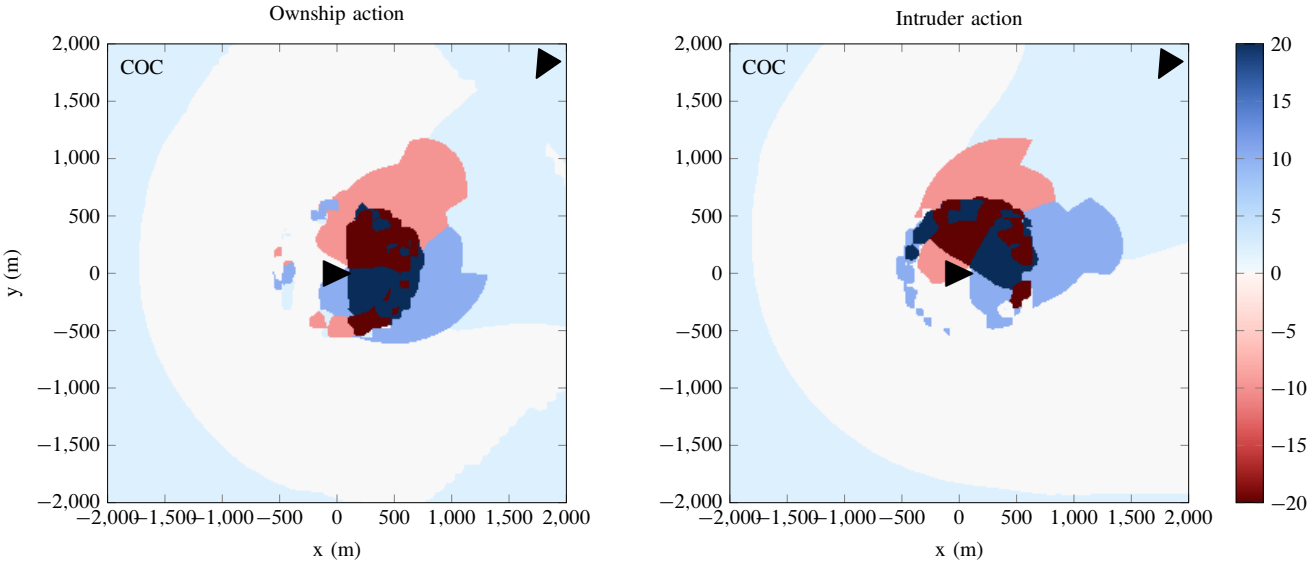
The solver was implemented in Julia, a high-level dynamic programming language for technical computing [19]. The solution was generated using a single core on a 3.40 GHz Intel i7 processor with 32 GB RAM. The procedure took approximately 12 hours to complete, with most of the work done in computing state transition probabilities. After this offline computation, the policy is consulted during operation in the form of a $U^\star(s, a)$ look-up table to select the best advisory to issue at each time step. The computation time to extract the best action from the table using (2) can be done in microsecond speed.

Figure 1 shows the policy for two encounter scenarios with the ownship aircraft state $(x, y, \psi, v) = (0\,\text{m}, 0\,\text{m}, 0°, 10\,\text{m/s})$ in the form of heat maps. In Figure 1a, the intruder is traveling towards the ownship aircraft with $180°$ relative heading at $10\,\text{m/s}$, and in Figure 1b, the intruder is traveling towards the ownship aircraft with $240°$ relative heading at $10\,\text{m/s}$. The heat maps show the action that would be issued for both aircraft when the intruder is at the plotted (x, y) coordinates with the indicated heading. Positive and negative values on the colorbars indicate left and right banks, respectively. The clear-of-conflict status actions are colored pale-blue to avoid confusion with the $0°$ bank advisory.

In general, the solution agrees well with intuition. For instance, consider the scenario where the intruder aircraft is at $(x, y) = (1000\,\text{m}, 500\,\text{m})$ heading straight towards the ownship aircraft. Here, Figure 1a shows that the optimal joint policy is for the ownship and intruder aircraft to both bank right. This combination of actions ensures that both aircraft turn away



(a) 180° head-on intruder conflict.



(b) 240° angled intruder approach.

Figure 1: Pairwise encounter MDP policy: Colors indicate bank angles for ownship and intruder with (x, y) relative coordinates and heading (black aircraft symbol). Pale blue region indicates aircraft is clear of conflict.

from one another. It is also important to note that the algorithm issues advisories between a set decision time period from a fixed set of advisories, which results in some surprising behavior. For instance, an interesting feature is the slight "indent" in the left inset of Figure 1a at roughly $(x, y) = (1500\,\text{m}, 0\,\text{m})$, which indicates that the ownship aircraft should wait to determine which way the intruder might turn before responding with a bank action. In Figure 1b the encounter scenario is identical except the intruder is flying at a relative heading of 240°. The nonzero banking region is rotated to prevent the intruder from flying into the exclusion region of the ownship aircraft from the latter's left. For both heading angles, once both aircraft experience loss of minimum separation (500 m) the system immediately advises both aircraft to bank more aggressively to resolve the conflict quickly.

5. Multi-threat Coordination

To coordinate multiple aircraft in conflict, this section uses MMDP decomposition, utility fusion, and online rewards to produce an approximate utility function for the global MMDP. As the search space for the approximate utility function can still be large (on the order of $|A|^n$ for n aircraft), we derive a set of centralized and distributed algorithms that solves the traffic management problem using a search heuristic.

A. Utility Fusion

To obtain a global MMDP solution from the pairwise subproblems, we need a measure of how good each individual pairwise solutions are when combined with other pairwise encounters. Utility fusion combines state-action values from subproblems to obtain a proxy to the full problem's state-action value function [8], [15]. It computes the utility $U_{ij}^{\star}$ for the pairwise encounter between two different aircraft i and j, assuming that the optimal policy for that encounter is followed in the future. For a set of pairwise encounter MDP states and actions

$$s = (s_1, \ldots, s_n), \quad a = (a_1, \ldots, a_n),$$

the best action for each aircraft can be found via a proxy global utility function $U^{\star}$ that results from fusion function f:

$$U^{\star}(s,a) = f\left(\left\{U_{ij}^{\star}\left(s_i, s_j, a_i, a_j\right) \mid i, j \in \{1, \ldots, n\},\ i < j\right\}\right).$$

Note that $U_{ij}^{\star}$ can include online rewards.

Our algorithm employs two fusion strategies. The first strategy defines f to be a summation:

$$f\left(\left\{U_{ij}^{\star}\right\}\right) = \sum U_{ij}^{\star},$$

which is also called the max-sum strategy ("max" here refers to us taking the argmax of the joint action space). Defining f in this fashion leads to counting nonzero bank angle costs multiple times, which would be reflected in the state-action utilities for each pairwise encounter. Since the system incurs the action cost multiple times when in reality the system can only alert once at each time step, the system is encouraged to delay issuing nonzero bank angles. This preference for later banking may be undesirable as it limits the amount of options available to each aircraft the closer they are to loss of minimum separation and can lead to more aggressive maneuvers at later times.

The second strategy defines f to be the minimum state-action utility over all pairwise encounters:

$$f\left(\left\{U_{ij}^{\star}\right\}\right) = \min U_{ij}^{\star}.$$

This method is also referred to as the max-min strategy. As opposed to the max-sum strategy, the max-min strategy avoids accumulating the cost of alerting for each pairwise encounter. This method also tends to provide more conservative policies in the sense that earlier banking is preferred.

To extract the joint policy, we compute

$$a^{\star} = \underset{a \in A^n}{\operatorname{argmax}}\, U^{\star}(s,a)$$

with a search heuristic that produces locally optimal solutions.

B. Online Rewards

The MMDP utility function is based only on the states and actions of the pairwise subproblems. Online rewards are introduced to incorporate preference for actions that are based on information not modeled in the subproblems and thus favor certain actions in real-time without introducing new state variables. Specifically, online rewards are used in our algorithm for coordinating joint aircraft maneuvers with fewer messages and thus less communication. During operation, the expected rewards calculated from the offline look-up table are added to the online rewards associated with that action.

C. Search Heuristic

As mentioned, the search space for joint policies for the approximate global utility function is on the order of $|A|^n$ and can thus be large for large numbers of aircraft n. To alleviate this issue, our technique finds a solution where each agent's policy is the best response to the policies employed by all other agents. The method relies on alternating maximization, which computes a policy for an agent that maximizes the joint reward while fixing the policies of other agents. The procedure is repeated until the joint policy converges to a local optimum, and is outlined in Algorithm 1.

Algorithm 1 Search heuristic

```
procedure SEARCH(f)                          ▷ fusion strategy f
    initialize: a* with COC status
    while a* not converged and not timeout do
        for each aircraft i do
            for each a^new ∈ A do
                a ← a*
                a_i ← a^new
                if f(a) > f(a*) then a* = a
    output: a*
```

D. Coordination Algorithm and Variants

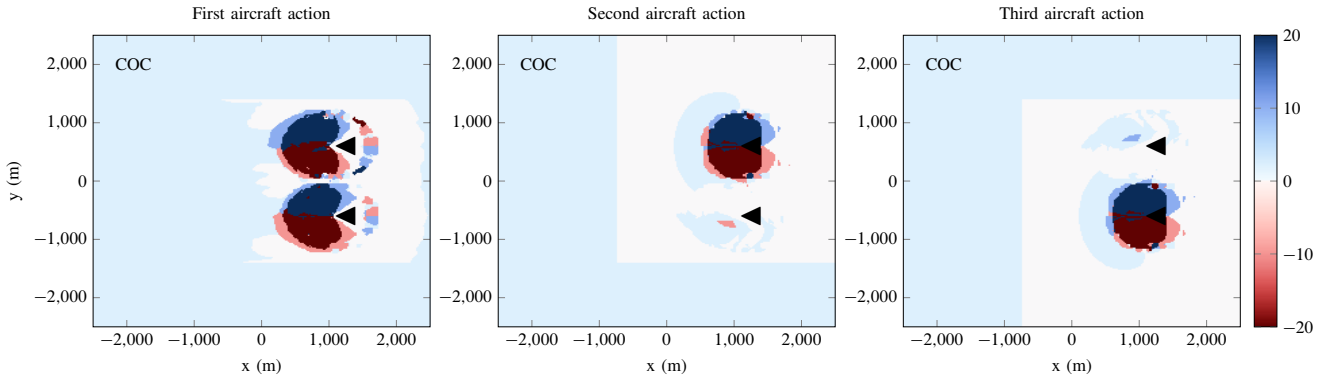
In the UAS Traffic Management system, aircraft information is updated frequently to a centralized system. To avoid over-estimating system specifications, we assume that the updates are simple and consist only of a unique aircraft identifier, global latitude and longitude, heading, and speed. These values can be directly mapped to our large MMDP and pairwise encounter relative states.

1) Centralized coordination: To incorporate the search heuristic into a serial coordination scheme, each aircraft first broadcasts its message to the system. The system then computes the locally optimal joint policy Algorithm 1, and sends resolution advisories to each aircraft individually. Figure 2 visualizes the joint policies computed with the max-sum and max-min utility fusion methods for a three-aircraft encounter scenario using a heat map. The first aircraft is flying straight with a heading of 0°, and the second and third aircraft are flying straight with headings of 180°. All aircraft are flying at 10 m/s. With the second and third aircraft coordinates fixed at (1200 m, 600 m) and (1200 m, −600 m), respectively, the first aircraft's coordinates are varied over $x, y \in [-2000\,\text{m}, 2000\,\text{m}]$. The left, center, and right insets plot the suggested action for the first, second, and third aircraft, respectively, when the first aircraft's coordinates are varied across the plot. As before, the colorbars indicate the bank angle for the advisories.

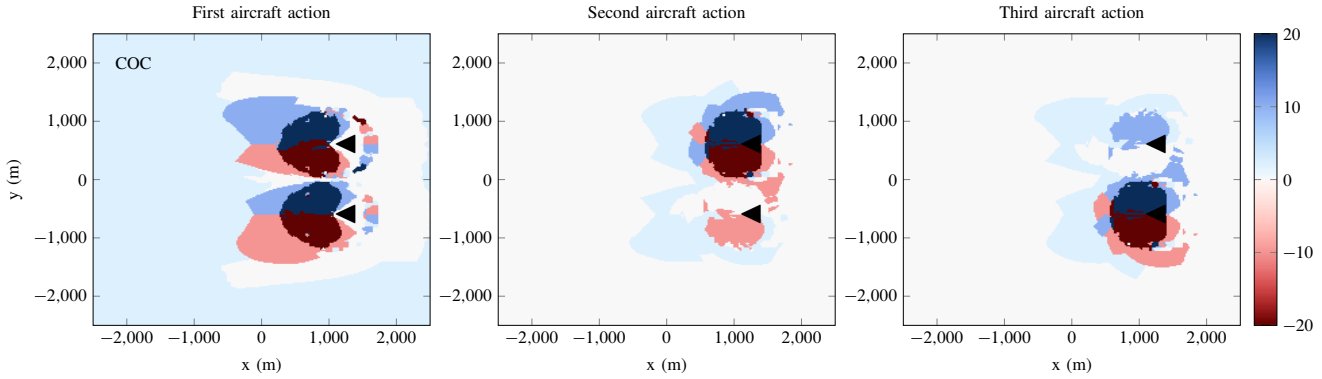
As explained in Section 5, max-sum utility fusion counts nonzero bank angle costs multiple times. This multiple counting thus produces the turn suggestion regions much smaller than the one generated by the max-min method. The "COC" status regions outside of the 4 km "fly-straight" zone on the max-sum plots are due to the limited range in the pairwise solution. An interesting feature from the max-min policy is the "patches" in the center and right insets. In the center inset, the pink patch at around $y = -1000$ m tells the second aircraft at $y = 600$ m to bank right if the first aircraft is in that patch. Likewise, the light blue patch in the right inset tells the third aircraft at $y = -600$ m to bank left if the first aircraft is in that patch. Intuitively, these patches indicate that both the second and third aircraft should synchronize their banking to avoid the first aircraft. One important property of the decomposition methods is that they do not begin advising nonzero bank angles (thus deviating aircraft from their intended trajectory) any earlier than the pairwise encounter policy. This behavior can be verified mathematically from the utility fusion definitions.

2) Distributed coordination I: The second coordination scheme is an obvious distribution of each iteration in Algorithm 1. Upon receipt of a potential conflict warning and the global state information from the central server, each aircraft initializes a COC joint policy and greedily computes the best action for itself assuming that all other aircraft will fly as if clear of conflict. The solutions are then broadcast and synchronized by the central server, producing a new joint policy with each entry being updated with the corresponding aircraft's greedily chosen action. The greedy computation is then executed again with the assumption that all other aircraft will execute the new joint policy, followed by a synchronization step. This step is repeated until convergence or time-out, wherein the best joint policy so far is executed by all aircraft. While simple in concept, we foresee significant communication costs due to synchronization steps at each iteration.

3) Distributed coordination II: This alternative distributed scheme attempts to minimize communications cost at the risk of having unsynchronized joint policies across aicraft. In this scheme, all aircraft in the encounter execute their own search using Algorithm 1 and combine their solutions to find a better overall joint policy. Conceptually, the multiple instances of search solutions serve as multiple random restarts of the local optimization scheme. As a result, the aircraft may end up with different joint policies, and we introduce an "approximate synchronization" procedure that uses only one joint



(a) Policy heat map with max-sum utility function.



(b) Policy heat map with max-min utility function.

Figure 2: Triple-aircraft encounter policy visualization: The color at any coordinate indicates suggested bank angles for the case where the first aircraft is at the coordinates flying with a 0° angle heading.

policy broadcast from each aircraft. This method encourages synchronization by incorporating these broadcast coordination messages as online rewards (in addition to utility fusion). But because it does not guarantee perfect synchronization—the cost of limiting communications—the performance might be negatively impacted as compared to the previous two coordination methods. The procedure is as follows.

Each aircraft first receives broadcast information from other aircraft in the encounter from the central system. They then independently compute their solutions to the global problem with the search heuristic. After computing their own joint policy solutions, they send coordination messages consisting of their solutions to the central server, which are then received by all aircraft. Next, each aircraft incorporates all messages. At this step, each aircraft assumes that the action corresponding to each intruder aircraft's own action in the broadcast joint policy is fixed. That is, if aircraft i broadcasts a joint policy stating that $a_i = 10°$, all other aircraft assume that aircraft i's action is held fixed at 10°. (2) for every joint policy broadcast, a reward is associated with the actions that other aircraft suggest an aircraft should do. In our experiments, positive rewards of 0.5 are added for each received suggested action that coincide with the ownship's action. For instance, if aircraft i and j both broadcast joint policies indicating that aircraft k should bank at $-10°$, the total positive reward of 1.0 would be incorporated into the utility function as a online reward for also taking action $-10°$. In the final step, a greedy search is executed by each aircraft for the best action it should take given fixed intruder bank angles and the new online rewards. These steps are summarized as follows:

1) initialize resolution
2) broadcast own state and receive other states
3) execute independent search heuristic
4) send and receive coordination messages
5) incorporate messages

6) find and execute greedy action.

6. Numerical Simulation

The speed and scaling of our algorithms are demonstrated with a range of examples randomly generated from an encounter model. These examples are intentionally kept simple for illustrative purposes, but can easily be extended with more refined networks and aircraft models. This section presents the encounter model and baselines against which the methods are benchmarked against. This section concludes with a discussion of simulation results.

A. Encounter Model

The coordination algorithms were evaluated against a stress-test set of multi-threat encounters randomly generated from an encounter model. As described in the previous section, the positions and velocities of each aircraft were available to them internally, but not to other aircraft. Although encounters between more than three aircraft are very rare for the conventional transport aircraft, a large part of current commercial interest surrounding unmanned aircraft stems from the potential to use a large number of transport drones.

Taking into account the above consideration, the multi-threat encounters ranged from two to ten aircraft. The positions were initialized uniformly randomly in an annulus such that they were not initially in conflict. Specifically, the annulus had inner and outer radii of 2000 m and 3000 m and if a new aircraft added is closer to some other aircraft than 600 m, we resample the new aircraft position to avoid initializing aircraft already in conflict. The speeds of the aircraft were uniformly randomly initialized between 10 m/s and 20 m/s, and the headings were initialized to always point straight towards the annulus center. This ensured that every encounter would have potential conflicts. In the future, we could develop a more sophisticated encounter model from, say, recorded radar data that is statistically representative of encounters between unmanned aircraft. At present, however, such data is not yet available at the level of what exists for commercial aircraft (e.g., TCAS data used for the encounter model in [20]).

B. Aircraft Model

An ODE solver was used for dynamics equations, with Gaussian noise in acceleration and banking. The aircraft maps the resolution advisories to PID control policies tuned to reach the target bank angle:

$$\ddot{\phi} = -2\omega_n \dot{\phi} + \omega n^2 \left(\phi^{\text{tgt}} - \phi\right),$$

where ω_n is a constant related to the amount of control power available and $a\phi^{\text{tgt}}$ is the target bank angle defined in the joint policy. In simulation, $\omega_n = 0.2$.

To model state uncertainty due to measurement error from, say, GPS inaccuracy, the simulations model the heading and speed error as zero-mean Gaussians with 2° and 1 m/s standard deviations, respectively. The aircraft longitude and latitude information are subject to zero-mean Gaussian noise with 50 m standard deviation. To account for bank angle command errors, the inputs are subject to a zero-mean Gaussian noise with 2° standard deviation in simulation.

When the state is partially observable, a probability distribution over the state space can be inferred from a recursive Bayesian estimation from a sequence of state measurements. This distribution is also called a belief state. In our algorithm, separate belief states are maintained for each UAS. To extract the policy from a belief state instead of an exact state, we use the QMDP solution $U^\star$ as follows:

$$\pi^\star(b_i) = \underset{a \in A}{\operatorname{argmin}} \sum_{s_i} b_i(s_i)\, U^\star(s_i, a),$$

where b_i is the belief distribution for the ith pairwise encounter and $b_i(s_i)$ is the probability of being in state s_i based on the belief distribution.

C. Baseline Methods

Our centralized and distributed coordination algorithms are tested against a simple command arbitration heuristic and uncoordinated algorithm that also uses utility fusion. The latter baseline is designed to demonstrate how well coordinated aircraft do against uncoordinated ones.

1) Closest threat command arbitration: In the first of our baseline algorithms, we consider the closest threat command arbitration method. This method separates an n-aircraft multi-threat scenario into n pairwise encounters for each of the n aircraft. Each aircraft then executes the action suggested by the solution to pairwise encounter with the lowest separation distance. Because the closest intruder is often the most immediate threat, prioritizing this pairwise encounter seems sensible.

2) Uncoordinated with utility fusion: In the second baseline, each aircraft is able to receive the basic state information for all other aircraft from a central system, but is unable to tell what other aircraft might do. Each aircraft then assumes that all other aircraft are white noise intruders that travel roughly along their initial headings. The action policy is extracted using utility fusion greedily with fixed zero bank angles for all other aircraft.

7. Results and Analysis

The experiments consist of 1,080,000 simulations with encounters ranging from two to ten aircraft running on a 3.40 GHz Intel i7 processor with 32 GB RAM. This section discusses the results.

A. Decision Time

The worst-case decision time for any algorithm was never more than 50 ms—much smaller than the time between decisions of 5 s. The decision time results for each algorithm using the max-min utility function are plotted in Figure 3, and the max-sum results are essentially identical. The legend is to be read as follows

- closest: closest-thread command arbitration
- uncoord.: uncoordinated with utility fusion
- centralized: centralized coordination
- dist. I: distributed coordination I
- dist. II: distributed coordination II.

B. Safety Performance

Figure 4 illustrates the performance of the baseline and coordination algorithms as the number of aircraft in potential conflict with each other is increased. The plot is the result of encounter simulations for both utility fusion strategies. It plots the probability that an encounter results in a loss of minimum separation between any pair of aircraft for different algorithms and utility fusions. For example, if there were two distinct pairs of aircraft that came into conflict for a four-aircraft scenario throughout its entire simulation, the number of conflict increases by two. In the same scenario, the total number of potential conflicts is $\binom{4}{2} = 6$. Each encounter was simulated for 500 seconds. The figure omits conflict probabilities for the closest-threat command arbitration method as conflicts occur an order of magnitude more frequently than in the distributed coordination methods.

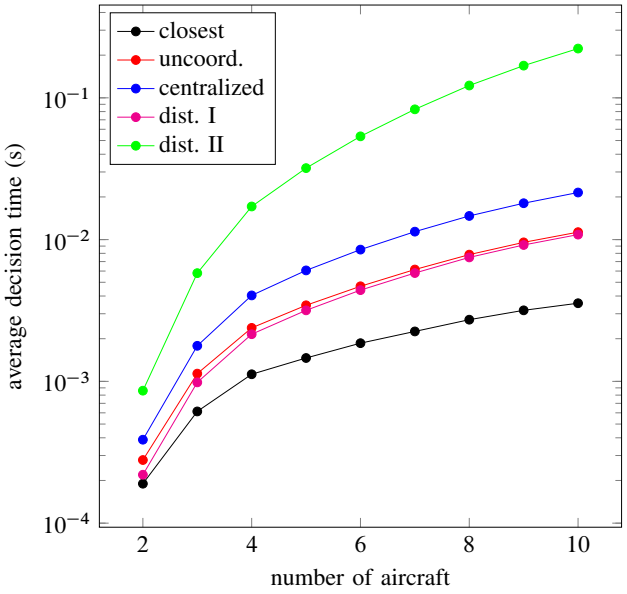
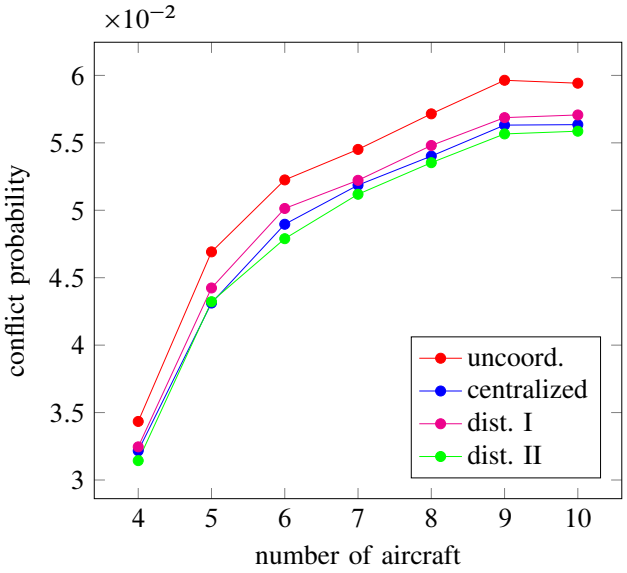


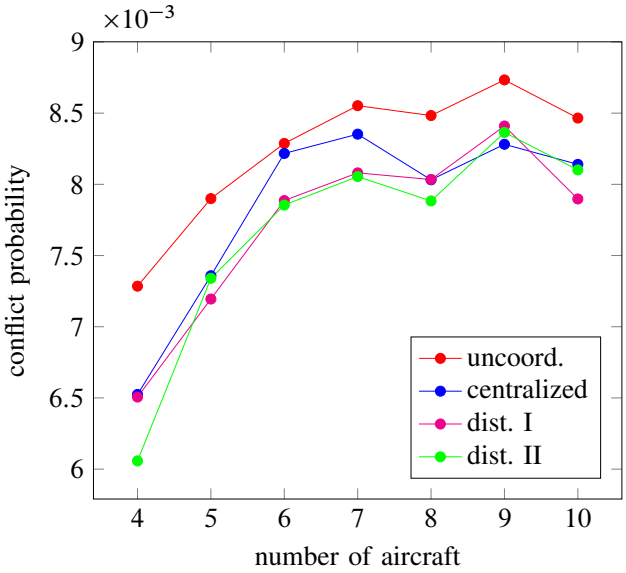
Figure 3: Average decision times for a family of encounters with max-min utility fusion.

It is not surprising that our methods performed much better than the closest-threat command arbitration method. Past work suggest that utilizing state-action values from sub-agents of a complex system can result in better performance than trying to arbitrate over actions from these sub-agents [9], [15], [21]. Even the uncoordinated baseline significantly outperformed the closest-threat baseline by five times for the max-sum strategy. As expected from the policy visualization and explanation, that the max-min strategy is much more conservative than the max-sum strategy improves the overall safety.

In all cases, the max-min utility fusion produced policies that were about an order of magnitude safer than the max-sum utility fusion policies. For any number of aircraft, the distributed coordination algorithms does about ten times as well as the closest-threat command arbitration heuristic, and about 10% better than the uncoordinated method. The two distributed schemes perform roughly as well as the centralized methods, which is surprising given that the first variant uses less compute time on average and the second variant does not guarantee the synchronization of joint policies. The results thus validate the effectiveness of both approaches. The second distributed coordination algorithm is more practical than the first due to the larger communication cost in the



(a) Conflict probability with max-sum utility function.



(b) Conflict probability with max-min utility function.

Figure 4: Conflict probability as the number of aircraft in encounter increases with max-sum utility fusion.

multiple synchronizations of the latter. However, the development of an asynchronous version of the first distributed scheme might be worthwhile to capitalize on its reduced computation requirements.

C. *Safety vs. Alert Rate Trade-Off*

To trade-off between safety (conflict probability) and alert rate, we vary the weight λ_4 on the reward component for the clear-of-conflict indicator function in (4). Specifically, we varied λ_4 on a uniform logarithmic scale from 10^{-3} to 10^{2}. The trade-off plot is shown in Figure 5, where the alert rate is defined to be the average number of alerts per aircraft per encounter simulation, and the conflict probability is defined to be the same as above. We only plot the curves generated by the centralized and distributed methods that use the max-min utility fusion method, since they were the highest performing algorithms. The best performing points are at the bottom-left "knee" of the Pareto frontier, minimizing both conflict probability and alert rate, and correspond to $\lambda_4 \in [1, 10]$. The plot also suggests that the centralized coordination algorithm is slightly better than other schemes on both the safety and alert rate metrics.

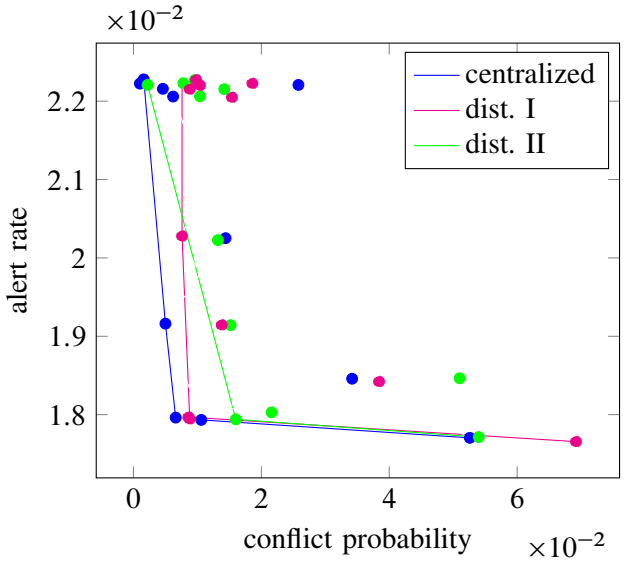


Figure 5: Trade-off plot for safety vs. alert rate. The points with the best performance are on the bottom-left "knee" of the curves.

8. Conclusion and Future Work

We have demonstrated three coordination algorithms that can solve the problem rapidly and effectively in the context of UAS Traffic Management. Joint policies that coordinate multiple aircraft turns can be computed from a look-up table and online reward functions in the order of microseconds. Future work will include more complex pilot response models that explicitly take into account noncompliance.

We will also explore an asynchronous version of the second distributed coordination scheme to alleviate potential communication issues. Because a priority for the traffic management system is to ensure safety while minimizing trajectory deviations for aircraft, these coordination algorithms need to be integrated with a return-to-path algorithm.

Acknowledgments

This research is sponsored by NASA under contract NAS2-03144 with UCSC (UARC). The authors wish to thank Bassam Musaffar and Heinz Erzberger from NASA Ames for their guidance at the beginning of this project. This work has also benefited from conversations with Claire Tomlin, Mo Chen, Qie Hu, Jaime Fernandez-Fisac, Casey D. Mackin, Zachary Sunberg, Eric Mueller, and Michael Owen.

Supplementary material

The software implementation of this work can be found together with documentation at

`https://github.com/sisl/ConflictAvoidanceDASC`.

References

[1] J. D'Onfro, "NASA drone traffic management system: This NASA project is the best hope to stop a potential drone disaster," *Business Insider*, Sep. 2014. [Online]. Available: http://www.businessinsider.com/nasa-drone-traffic-management-system-2014-9.

[2] T. Schouwenaars, M. Valenti, E. Feron, and J. How, "Implementation and flight test results of MILP-based UAV guidance," in *IEEE Aerospace Conference*, 2005.

[3] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[4] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[5] H. Y. Ong and J. C. Gerdes, "Cooperative collision avoidance via proximal message passing," in *American Control Conference (ACC)*, 2015.

[6] H. Erzberger, T. Lauderdale, and Y. Chu, "Automated conflict resolution, arrival management, and weather avoidance for air traffic management," in *International Congress of the Aeronautical Sciences*, 2010.

[7] H. Erzberger and K. Heere, "Algorithm and operational concept for resolving short-range conflicts," *Journal of Aerospace Engineering*, vol. 223, pp. 225–243, 2009. DOI: 10.1243/09544100JAERO546.

[8] M. Kochenderfer and J. Chryssanthacopoulos, "Robust airborne collision avoidance through dynamic programming," MIT Lincoln Laboratory, Project Report ATC-371, 2011.

[9] J. P. Chryssanthacopoulos and M. J. Kochenderfer, "Decomposition methods for optimized collision avoidance with multiple threats," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 2, pp. 398–405, 2012.

[10] M. P. Owen, R. Williams, and A. Mezhirov, "Robust UAS separation using Markov decision processes," in *Digital Avionics Systems Conference (DASC)*, 2015.

[11] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," vol. 1, no. 4, pp. 179–189, 2000.

[12] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, "Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[13] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[14] C. A. Rothkopf and C. Dimitrakakis, "Preference elicitation and inverse reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases*, Springer, 2011, pp. 34–48.

[15] S. J. Russell and A. Zimdars, "Q-decomposition for reinforcement learning agents," in *International Conference on Machine Learning (ICML)*, 2003.

[16] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[17] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *International Conference on Machine Learning (ICML)*, 1995.

[18] M. Hauskrecht, "Value-function approximations for partially observable Markov decision processes," vol. 13, pp. 33–94, 2000.
[19] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," *ArXiv preprint arXiv:1209.5145*, 2012.
[20] T. B. Billingsley, L. P. Espindle, and J. D. Griffith, "TCAS multiple threat encounter analysis," MIT Lincoln Laboratory, Project Report ATC-359, 2009.
[21] J. K. Rosenblatt, "Optimal selection of uncertain actions by maximizing expected utility," *Autonomous Robots*, vol. 9, no. 1, pp. 17–25, 2000.