An Efficient Algorithm for Multiple-Pursuer-Multiple-Evader Pursuit/Evasion Game

Joshua R. Bertram
Iowa State University
Ames, IA 50011
bertram1@iastate.edu

Peng Wei
George Washington University
Washington, DC 20052
pwei@gwu.edu

We present a method for pursuit/evasion from the Artificial Intelligence community that is highly efficient and and scales to large teams of aircraft. The underlying algorithm, FastMDP, is an efficient algorithm for solving Markov Decision Processes (MDPs) that supports fully continuous state spaces. We demonstrate the algorithm in a team pursuit/evasion setting in a 3D environment and study performance by varying sizes of teams up to 100 vs 100. We show that as the number of aircraft in the simulation grows, computational performance remains efficient and is suitable for real-time systems. To the authors knowledge this is the first approach for large scale pursuit-evasion problems that remains efficient enough to be deployed on embedded hardware. We use probability-to-win and survivability metrics that describe the teams' performance over multiple trials to show that the algorithm performs consistently. We provide numerical results showing control inputs for a typical 1v1 encounter and provide videos for 1v1, 2v2, 3v3, 4v4, and 10v10 contests to demonstrate the ability of the algorithm to adapt seamlessly to complex environments.

I. Introduction

Pursuit/evasion games pit two opponents against each other such that the pursuer must capture the evader. Within the aerospace community, pursuit/evasion of aircraft has long been of interest and is seeing a resurgence of interest due to a growing capability and acceptance of autonomous unmanned aircraft. Additionally, pursuit/evasion games are interesting in that they pose scalability challenges especially to UAV swarm applications. Problem formulations which lead to efficient and effective pursuit/evasion for 1 versus 1 (1v1) contests do not always allow efficient formulation with larger contests with multiple members per team (e.g., 2v2, 10v10). For problem formulations and algorithms that can support larger teams, it may be possible to solve the problem offline, but it may be exponentially harder and challenging in an online manner.

In this paper we propose a pursuit/evasion problem formulation based on Markov Decision Processes (MDPs) and use our recently proposed FastMDP algorithm [2, 3] to efficiently solve the problem for large teams up to 100 vs 100. MDPs are used to solve problems with complex dynamics and systems of reward. The algorithm seamlessly switches between pursuit and evasion while simultaneously avoiding collisions with other aircraft in the same team and with the ground. The algorithm is adaptable to multiple aircraft types through the use of forward projection of the aircraft dynamics. The aircraft dynamics model is modular and can be replaced with other models as needed.

Our main contributions for this work are:

- Pursuit evasion in a 3D continuous state space based off a Markov Decision Process formulation;
- Method that does not require any offline training phase that solves the problem online;
- Forward projection module that allows the algorithm to support any arbitrary aircraft type;
- Demonstration of efficient algorithm performance that scales to large team sizes up to 100 vs 100.

We additionally develop a 3D visualization tool to evaluate the algorithm and to provide insight on the complexity of the problem.

In Section II we identify and discuss related work. In Section III we briefly provide background on Markov Decision Processes. In Section IV we describe the method we use, including the aircraft dynamics, as well as the details of the

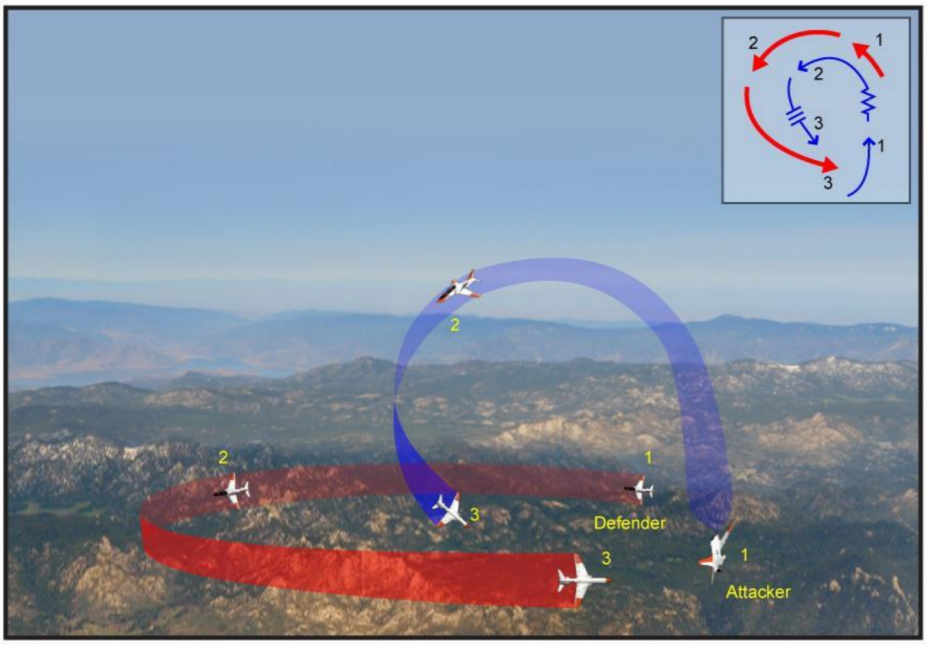


Fig. 1 Example of a high yo-yo maneuver from public domain CNATRA [1] training manual.

Markov Decision Process problem formulation used in this approach. Section V describes the experimental setup to evaluate the pursuit/evasion contests, including the definition of two metrics we use to evaluate the behavior. Section VI describes the results of our experiments demonstrating the efficient performance and consistent behavior of the algorithm. Section VI also provides links to videos showing examples of varying sized teams competing against each other.

II. Related Work

There is extensive work from many communities which address different approaches to pursuit/evasion. We describe several approaches and discuss how they relate to Markov Decision Process approach used in this paper.

Eklund etc. [4] described a nonlinear model predictive control (NMPC) approach to a pursuit/evasion problem using a set of cost functions with repulsive and attractive natures to shape the behavior of the pursuer. An iterative optimization method was used to produce a solution at each time step using simplified aircraft dynamics. Multiple matrices in the NMPC formulation required tuning to obtain good behavior. It is worth noting that the cost functions used in their work are analogous to reward functions used for Markov Decision Processes.

Schopferer and Pfeifer [5] proposed a method to perform flight planning in the presence of a uniform wind field, with the aircraft motion modeled with trochoids. The three dimensional flight path is constructed by superimposing a horizontal and vertical solution to obtain an approximate 3D path. A probabilistic roadmap planner is used to generate global plans.

Vector field approaches have also been used for pursuit/evasion problems. Goncalves etc. [6] described a vector field approach for convergence, circulation, and correction around a closed loop pattern. Lawrence etc. [7] presented a vector field approach for circular (or warped circular) patterns, and also describes a switching mechanism to handle waypoint following or arbitrary paths. Stable tracking of the vector field is explored using Lyapunov techniques. Vector fields can be viewed as similar in nature to the optimal policy that is generated by solving a Markov Decision Process. Where vector fields are generally applied over a continuous state space, MDP optimal policies normally describe actions that are intended to cause a transition from the current discrete state to a desired next discrete state.

Within the robotics and computational geometry community, pursuit/evasion is often considered in a different context. The pursuer(s) are attempting to search through an environment to observe the evader(s), similar to security guards searching through a museum for a potential intruder. Often in these problem formulations, the goal is identifying the minimum number of pursuers needed in order to guarantee that if an evader is present within the environment that it will be detected, and is not focused on tracking or chasing the evader as in the target problem of this paper. However,

these works are instructive as the algorithm used in this paper is built on the recognition that an MDP can be represented as a graph. Examples of this type of pursuit/evasion problem are [8–10]. An example of graph based pursuit/evasion problem applied to graphs of infinite nodes is [11], where they describe the problem as a cop-and-robbers problem and define a winning strategy as preventing the robber from visiting a node in the infinite graph infinitely many times. This allows strategies which either catch the robber or force the robber to flee 'to infinity'. Markov Decision Processes are normally viewed as a tree of sequential actions, but can also be understood as a graph. As most MDP problems normally have a discrete state space, this graph would normally also have a finite number of nodes. Our method provides a way to support MDP problem formulations with continuous state spaces, and the corresponding graph would then have an infinite number of nodes. Like the cop-and-robbers problem above, forcing an adversary to flee would be an acceptable strategy for our aircraft pursuit/evasion problem as well.

Jia etc. [12] proposed a continuous-time Markov Decision Process (CTMDP) approach where variable time steps are allowed to be taken within a discretized state space where the transition function is defined instead as a transition rate function, allowing the possible resulting state transitions to be predicted with varied time steps. The large state space is simplified by classifying the states into neutral, advantaged, disadvantaged, and mutually disadvantaged categories and a Bayesian method is used to determine the transition probabilities. Pursuit/evasion within a 2D grid world environment is considered.

Within the optimal control community, one area of related work is Differential Dynamic Programming (DDP) which uses dynamic programming to iteratively improve a local optimal control policy. Sun etc. [13] used DDP to solve an adversarial aircraft pursuit/evasion problem, terming their approach as game-theoretic DDP (GT-DDP) by combining DDP with a min-max problem formulation. Differential Dynamic Programming and Markov Decision Processes have much in common and both stem from Bellman's original work on dynamic programming [14]. Where the optimal control field focuses on the Hamilton-Jacobi-Bellman (HJB) equation and differentiable dynamics, MDPs often generalize the dynamics into a (deterministic or stochastic) transition function which captures uncertainty about the environment through probabilities (similar to those used for Markov chains.) Comparing [13] to this paper's work, GT-DDP in [13] does have a much richer capability to incorporate system dynamics, but this comes at the expense of additional computation time and a need for convergence of the iterative nature of the algorithm.

The most relevant paper to this work is [15] which describes a Markov Decision Process based pursuit/evasion problem for aircraft using approximate dynamic programming. A state space was formed from a set of features which minimized mean squared error using a forward-backward search. Trajectory sampling was used to obtain training data that would be likely to have value during training. Reward shaping was used to guide the exploration to the desired behavior in the form of a scoring function heuristic developed by an expert. Rollout was used to extract a refined policy from the approximation computed via approximate dynamic programming (ADP) and was accelerated with a neural net. The dynamics model for the airplane used is a Dubin's airplane without any vertical components or altitude modeled.

There are some subtle differences between this paper and the work in [15]. [15] is a good example of using a variety of practical techniques to deal with the intractability of large MDP state spaces, whereas this work explicitly uses a state space designed to be intractable by traditional MDP methods via the use of a continuous state space resulting in an MDP with an infinite number of states in order to demonstrate scaling to continuous state spaces. [15] uses a 2D aircraft model, where this paper uses a 3D model to demonstrate scaling to a continuous 3D state space and to demonstrate full maneuvering by the aircraft (e.g., loops, rolls, spirals). In this paper, no reward shaping is required to speed up or aid convergence, as the underlying MDP is solved directly without relying on typical methods used for approximate dynamic programming. And finally, in [15] 1v1 pursuit/evasion is explored where in this paper scaling to 100 v 100 teams is demonstrated.

In [16], a pursuit evasion problem is studied from an optimal control perspective using pseudospectral methods. A controller is developed which achieves approximately a 3Hz rate on desktop-class hardware for a 1v1 optimal evasion.

Also of note are [17] and [18]. Park et al [17] used a higher fidelity 3D model and a min-max approach over a sliding window to demonstrate 1 vs 1 pursuit/evasion, and while the behavior in simulation appears promising, the real-time performance of the algorithm is not reported. In [18], a reinforcement learning approach is taken using deep Q-learning using a 2-layer multi-layer perceptron as the function approximator, and with a modified epsilon-greedy exploration strategy where a heuristic function used in place of random action in order to avoid wasteful actions during exploration. Performance is examined in 2D.

III. Background

Markov Decision Processes (MDPs) are a framework for sequential decision making with broad applications to finance, robotics, operations research and many other domains [19]. MDPs are formulated as the tuple (S, A, R, t) where $s_t \in S$ is the state at a given time t, $a_t \in A$ is the action taken by the agent at time t as a result of the decision process, $r_{t+1}(s_t, a_t)$ is the reward received by the agent as a result of taking the action a_t from s_t, and $T(s_t, a, s_{t+1})$ is a transition function that describes the dynamics of the environment and capture the probability $P(s_{t+1}|s_t, a_t)$ of transitioning to a state s_{t+1} given the action a_t taken from state s_t.

A policy π can be defined that maps each state $s \in S$ to an action $a \in A$. From a given policy $\pi \in \Pi$ a value function $V^\pi(s)$ can be computed that describes the expected return that will be obtained within the environment by following the policy π. The value function can be expressed in the iterative Bellman equation as follows, where $r_{t+1}(s_t, a_t)$ represents immediate reward collected by taking an action a_t from state s_t and a value of $V(s_{t+1})$. This the value function for any state is the current reward plus the discounted future reward that can be obtained by taking the best action from the current state, and is an expectation of the future reward that can be obtained from the current state.

$$V(s) = \max_{a_t \in A} \left[r_{t+1}(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} T(s_{t+1}|s_t, a_t) \cdot V(s_{t+1}) \right] \tag{1}$$

The solution of an MDP is termed the optimal policy π^*, which defines the optimal action $a^* \in A$ that can be taken from each state $s \in S$ to maximize the expected return. From this optimal policy π^* the optimal value function $V^*(s)$ can be computed which describes the maximum expected value that can be obtained from each state $s \in S$. And from the optimal value function $V^*(s)$, the optimal policy π^* can also easily be extracted.

IV. Method

We use the algorithm described in [2, 3, 20] as the underlying guidance and collision avoidance algorithm. The algorithm is modified to support a pursuit evasion problem and the aircraft dynamics model used allows complex aerial maneuvers which respect energy management trade-offs. Demonstration of scaling is shown with large teams performing pursuit/evasion together.

A. Aircraft Dynamics Model

The aircraft kinematic model is a 6 degree of freedom (6DOF) model which approximates fixed wing aircraft motion given inputs similar to stick and throttle inputs. The model provides a way to study the algorithm's behavior without requiring full aerodynamics to be modelled. The algorithm needs this 6DOF model to provide "forward projection". This means that from a given current state, the model must be able to calculate the future state of applying a given set of possible control actions for a fixed number of time steps. Any model which satisfies this requirement can be integrated with the algorithm, including full-fidelity 6DOF fixed-wing models, helicopters, quad rotors, and models with underlying autopilot controllers.

The model used is an extension of the 6DOF formulation in [17] and also incorporates a few additional terms from the model in [21]. It should be considered as a simplified model of [21]. Variable definitions are:

- n_x: Throttle acceleration directed out the nose of the aircraft in g's,
- V: Airspeed in meters/second,
- γ: Flight path angle in radians,
- x, y, z: position in NED coordinates in meters where altitude $h = -z$,
- ϕ: Roll angle in radians,
- ψ: Horizontal azimuth angle in radians,
- α: Angle of attack in radians with respect to wind reference frame.

The inputs to the model are: (1) the thrust n_x, (2) the rate of change of angle of attack $\dot{\alpha}$ and (3) the rate of change of the roll angle $\dot{\phi}$.

The equations of motion for the aircraft are:

$$\dot{V} = g\left[n_x \cos\alpha - \sin\gamma\right], \tag{2}$$

$$\dot{\gamma} = \frac{g}{V}\left[n_f \cos\phi - \cos\gamma\right], \tag{3}$$

$$\dot{\psi} = g\left[\frac{n_f \sin\phi}{V\cos\gamma}\right], \tag{4}$$

where the acceleration exerted out the top of the aircraft n_f in gs is defined as:

$$n_f = n_x \sin\alpha + L, \tag{5}$$

with a lift acceleration of $L = 0.5$. Here, 1 “g" is a unit of acceleration equivalent to 9.8 m/s^2. L was chosen to provide some amount of lift while in flight to partially counteract gravity and provide a stable flight condition with a low positive α angle of attack in the 6DOF model. For a true aerodynamic model, this lift varies by the velocity (Mach number), but this level of detail is omitted in our simplified 6DOF.

The kinematic equations are:

$$\dot{x} = V\cos\gamma\cos\psi \tag{6}$$

$$\dot{y} = V\cos\gamma\sin\psi \tag{7}$$

$$\dot{z} = V\sin\gamma. \tag{8}$$

While this model is not aerodynamically comprehensive, it is sufficient to describe aircraft motion suitable for examining the algorithm behavior without loss of generality. Again, our algorithm can integrate with any aircraft dynamics model.

1. Forward Projection

In order to determine the future state resulting from a given action, we use forward projection to simulate the dynamics forward in time over a receding horizon. We use a time step $dt = 0.1$ seconds and a window $W = 1.0$ seconds to compute the result of taking an action a over the window W. After selecting an action and applying it to the simulation, we advance the simulation one time step (0.1 seconds). Thus an action is chosen at a 10 Hz rate with a 1 second forward projection receding horizon.

The simulated future states can be viewed as an approximation of the reachable states R_s, and are applied to the solution of the Markov Decision Process (MDP) to determine the value of the potential future states the agent might reach. Thus the agent follows the optimal policy of the MDP at each time step by determining which future reachable state is most valuable, and then takes the action in the next time step that will lead it towards that state. We realize that this is a very rough sampling of the full reachability set R_s, but the intent is to provide enough sampling of the MDP value function to assess the gradient and only a rough sampling is required for this purpose. We also acknowledge other methods may be used that may be more efficient but we leave this to future work.

Each team is provided with different aircraft performance limits which serve to provide the “blue" team (team 0) with a performance advantage over the “red" team (team 1) and prevents deadlocks where neither team is able to obtain an advantage over the other. Table 1 lists the performance limits, where the speed of sound $Mach = 343\ m/s$. These limits were chosen to represent a highly maneuverable subsonic UAV and do not represent any real aircraft.

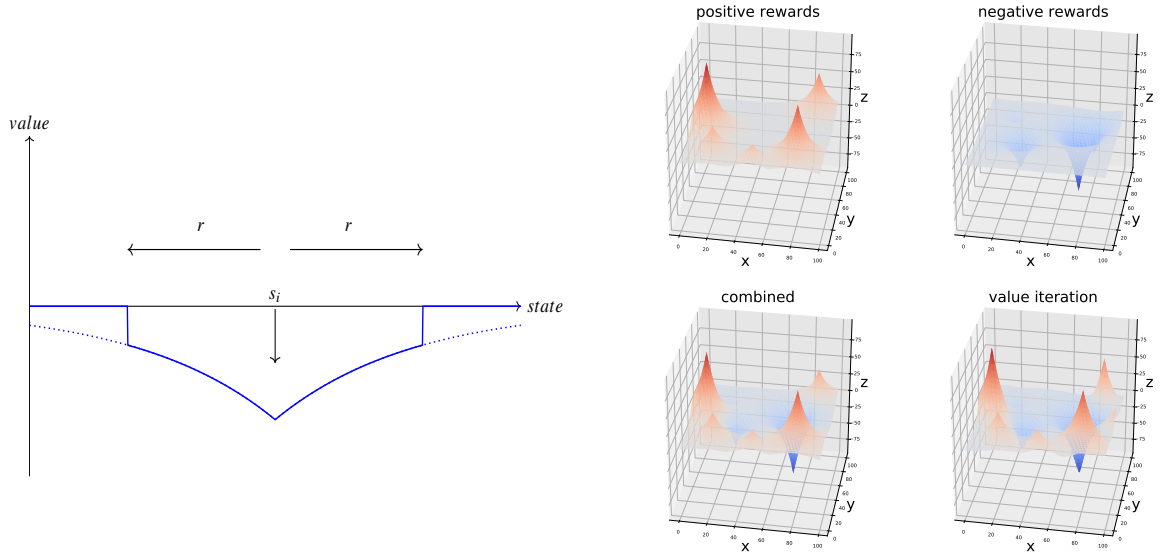
B. MDP Formulation

1. State Space

We define the environment where the aircraft operate as a flat-earth tangent plane which is treated as a continuous state space. There are two teams of aircraft in this environment: a “blue" team and a “red" team. Each aircraft (an “ownship”) is controlled by our proposed algorithm.

The state includes all the information each ownship needs for its decision making: the full aircraft state of the ownship, the position and velocity of every teammate aircraft, and the position and velocity of every opponent aircraft.

Each ownship is aware of its own aircraft state produced by aircraft dynamics model. For each ownship, the state is formed by concatenating the following:



(a) A risk well showing exponential decay of a negative reward out to a fixed radius beyond which the negative penalty is truncated.

(b) FastMDP solving positive and negative rewards, combining the results, and comparing to the solution produced by the value iteration algorithm traditionally used to solve MDPs.

Fig. 2 FastMDP solves MDP using peaks that represent positive and negative rewards

Table 1 Limits on aircraft performance for each team

Team	V_{min} (Mach)	V_{max} (Mach)	$\dot{\psi}_{\min}$ (rad/s)	$\dot{\psi}_{\max}$ (rad/s)	$\alpha_{\min}$ (rad)	$\alpha_{\max}$ (rad)
Blue	0.1	0.35	-1.5	1.5	-.009	.69
Red	0.1	0.30	-1.3	1.3	-.009	.52

- ζ the ownship aircraft state: $x, y, z, V, \psi, \phi, \gamma, \alpha$,
- for each teammate aircraft $j \in J$: $f_j = \{$ the position $f_{j,x}, f_{j,y}, f_{j,h}$ and velocity $f_{j,v_x}, f_{j,v_y}, f_{j,v_h}\}$, and
- for each opponent aircraft $k \in K$: $i_k = \{$ the position $i_{k,x}, i_{k,y}, i_{k,h}$ and velocity $i_{k,v_x}, i_{k,v_y}, i_{k,v_h}\}$

$$s_o = [\zeta, f_1, \cdots, f_j, i_1, \cdots, i_k] \tag{9}$$

where J represents the set of teammate aircraft, and K represents the set of opponent aircraft.

2. Action Space

Inputs to the model are (1) the rate of change of angle of attack $\dot{\alpha}$, (2) the rate of change of the roll angle $\dot{\phi}$, and (3) the axial force n_x.

The action space is then:

$$A = \{\dot{\alpha}, \dot{\phi}, n_x\}. \tag{10}$$

There are two teams of aircraft $k \in \{0, 1\}$ where team $k = 0$ is the “blue team" and $k = 1$ is the “red team". When the teams’ aircraft have equivalent performance, simulations often result in a stalemate which represent a Nash

equilibrium where neither aircraft is able to gain advantage over the other. In these cases, simulation will not naturally terminate. Therefore, in the simulations we provide a performance advantage to the blue team which more naturally leads to simulations that terminate.

Table 2 Action choices for each team

Team	$\dot{\phi}$ (rad/s)	$\dot{\alpha}$ (rad/s)
Red	-1, -.8, $\cdots$, .8, 1	-.5, -.4, $\cdots$, .4, .5
Blue	-1.5, -1.2, $\cdots$, 1.2, 1.5	-.5, -.4, $\cdots$, .4, .5

The primary mechanism to control the behavior of an agent in a Markov Decision Process (MDP) is through the Reward Function. The reward function provides both positive and negative reward to the agent. The optimal control community will likely find this concept foreign, as optimal control approaches seek to minimize (or maximize) a cost function which only contains positive terms such that a global minimum exists. MDPs however are from the artificial intelligence community where they seek to model a variety of problems which may not easily be adapted to the optimal control form. This leads to complex behaviors and problems that do not have an equivalent optimal control formulation. Very simple MDP formulations with only positive or only negative rewards may be able to be mapped to a corresponding optimal control problem, but this represents a small fraction of MDPs. The solution of an MDP maximizes the expectation of future reward, so in that sense it is an optimization problem. Whether to use an MDP or a traditional optimal control method is a trade-off, as each approach and problem formulation has advantages and disadvantages.

In our pursuit evasion problem, we will use positive and negative rewards that are coupled together to create tension between actions that the agent could potentially take. For example, we will place a positive reward near the location of an aircraft to attract other aircraft, but we will also place a negative reward at the aircraft to prevent a collision. A natural equilibrium develops between these positive and negative rewards that generates the desired behavior of approaching another aircraft without colliding with it.

Following the approach used in [20], we will treat each negative reward as a “risk well", which is a region of negative reward (i.e., a penalty) which is more intense at the center and decays outward until a fixed radius is reached, whereafter no penalty is applied. See Figure 2a for an illustration.

We present our reward function in terms of the behaviors we wish to obtain in Table 3. In this table, $\hat{p}$ represents the current position of an aircraft (teammate or opponent) and $\hat{v}$ represents that aircraft's current linear velocity. In some cases we project the aircraft's position forward in time with an expression $\hat{p} + \hat{v}t$ and then define a range of time as in $\forall t \in \{0, 1, 2\}$ to indicate that we create a reward at the location of the aircraft at each time step in the future indicated by the range of t.

All aircraft also receive a penalty below a certain altitude which prevents the aircraft from plummeting into the terrain. For this paper, h_{max} is the maximum height of the terrain that is loaded into the simulation. We define a minimum safe altitude known as the “hard deck" in which we will allow the aircraft to fly. Any aircraft which goes below the hard deck for the purposes of the game has crashed and is removed from the simulation. We define the hard deck $h_{\text{deck}} = h_{\text{max}} + 1000$. For any state with an altitude of h from the hard deck up to an altitude of $h_{\text{penalty}} = h_{\text{deck}} + 1000$, a penalty is applied $r_{\text{penalty}} = -(10000 - h)$ which is a very strong negative reward that will override any other positive rewards in the game.

C. Algorithm

We show the psueducode for the FastMDP algorithm in Algorithm 1. The algorithm at each iteration builds peaks from rewards in the environment, performs forward projection to determine a sampling of the reachability set, computes value at each reachable state from positive peaks, negative peaks, and a hard-deck penalty, and identifies the most valuable action. See Figure 3 for a high level overview of the algorithm.

Algorithm 1 FastMDP Algorithm

procedure FastMDP(*aircraftState*, *worldState*)
  $\mathbf{S_t} \leftarrow \mathbf{S}_0$ // randomized initial aircraft states
  $\mathbf{A} \leftarrow$ *aircraft actions (precomputed)*
  $\mathbf{L} \leftarrow$ *aircraft limits (precomputed)*
  $\mathbf{S}_{t+1} \leftarrow$ allocated space
  while aircraft remain **do**
    for each aircraft **do**
      $s_t \leftarrow \mathbf{S}_t[aircraft]$
      // Build peaks from rewards in the environment
      $\mathbf{P}^+ \leftarrow$*build pos rewards*
      $\mathbf{P}^- \leftarrow$*build neg rewards in Standard Positive Form*
      $\mathbf{P}^* \leftarrow$*build neg rewards for terrain in Standard Positive Form*
      // Perform forward projection
      $\mathbf{\Delta_1} \leftarrow fwdProject(s_t, \mathbf{A}, \mathbf{L}, 0.1\ s)$
      $\mathbf{\Delta_{10}} \leftarrow fwdProject(s_t, \mathbf{A}, \mathbf{L}, 1.0\ s)$
      // Compute the value at each reachable state
      $\mathbf{V}^* \leftarrow$ *allocate space for each reachable state*
      for $s_j \in \mathbf{\Delta_{10}}$ **do**
        // First for positive peaks
        for $p_i \in \mathbf{P}^+$ **do**
          // distance
          $d_p \leftarrow \left\| s_j - \textbf{location}(p_i) \right\|_2$
          $r_p \leftarrow \textbf{reward}(p_i)$
          $\gamma_p \leftarrow \textbf{discount}(p_i)$
          $\mathbf{V}^+(p_i) \leftarrow |r_p| \cdot \gamma_p^{d_p}$
        $V^+_{max} \leftarrow \max_{p_i} \mathbf{V}^+$
        // Next for negative peaks (in Standard Positive Form) including terrain
        for $n_i \in \{\mathbf{P}^-, \mathbf{P}^*\}$ **do**
          //distance
          $d_n \leftarrow \left\| s_j - \textbf{location}(n_i) \right\|_2$
          $r_n \leftarrow \textbf{reward}(n_i)$
          $\gamma_n \leftarrow \textbf{discount}(n_i)$
          // within risk well radius?
          $\rho_n \leftarrow negDist_i < \textbf{radius}(n_i)$

          $\mathbf{V}^-(p_i) \leftarrow int(\rho_n) \cdot |r_n| \cdot \gamma_n^{d_n}$
        $V^-_{max} \leftarrow \max_{p_i} \mathbf{V}^-$
        // Hard deck penalty
        if $\textbf{altitude}(s_t) < penaltyAlt$ **then**
          $V_{deck} \leftarrow 10000 - \textbf{altitude}(s_t)$
        else
          $V_{deck} \leftarrow 0$
        $\mathbf{V}^*[s_t] \leftarrow V^+_{max} - V^-_{max} - V_{deck}$
      // Identify the most valuable action
      $a^* \leftarrow \arg\max_s(\mathbf{V}^*)$
      // For illustration, the corresponding value
      $maxValue \leftarrow \mathbf{V}^*[a^*]$
      // And the next state when taking the action
      $\mathbf{s_{t+1}} \leftarrow \mathbf{\Delta_1}[a^*]$
      $\mathbf{S}_{t+1}[aircraft] \leftarrow \mathbf{s_{t+1}}$
    // Now that all aircraft have selected an action, apply it
    $\mathbf{S_t} \leftarrow \mathbf{S}_{t+1}$

Table 3 Rewards created for each ownship

For each teammate:

Magnitude	Decay factor	Location	Radius	Time steps	Comment
−100	.97	$\hat{p} + \hat{v}t$	$150 + 10t$	$\forall t \in \{0, 1, 2, 3, 4, 5\}$	Collision avoidance, 5 rewards
10	.999	$\hat{p}$	∞	N/A	Weak formation flight or clustering

For each opponent:

Magnitude	Decay factor	Location	Radius	Time steps	Comment
−300	.99	$\hat{p} + \hat{v}t$	$\hat{v}t$	$\forall t \in \{0, 1, 5, 10\}$	Collision avoidance, 4 rewards
200	.999	$\hat{p}$	∞	N/A	Pursuit

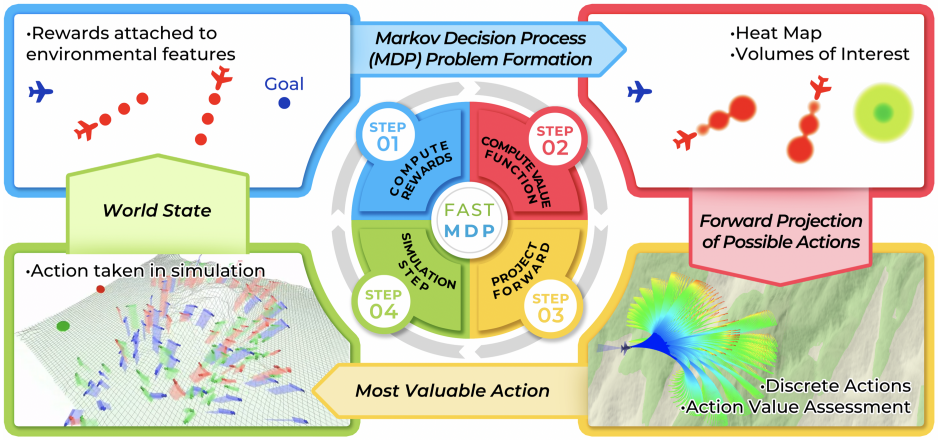


Fig. 3 High level operation of the FastMDP algorithm.

V. Experimental Setup

We demonstrate this MDP based planner in a 3D aircraft simulation showing a view of the two teams of aircraft. The simulation covers a configurable sized volume which contains a configurable number of team members on each of the two teams. Simulation begins with both teams spawned randomly on opposing sides of the environment. The teams must each avoid collisions with teammates while simultaneously pursuing members of the opposing team using only the reward system we have defined above.

At each time step, the simulation generates the state updates for each ownship. Each aircraft is controlled by an independent instance of the FastMDP algorithm, and at each time step each agent formulates an independent MDP, solves it, and takes an action. Each ownship forward projects each possible action over the receding horizon window of 1 second, and then uses the solution of the MDP to determine which action results in the highest valued future state. The action selected with this method will then be applied in simulation for 1 time step (0.1 seconds). The actions of all aircraft from both sides are selected and performed simultaneously without knowing the selected actions of any other aircraft in the simulation. In this paper, perfect information is provided to each aircraft and no uncertainty is considered. (Dealing with uncertainty is left to future work.) Simulation then advances by one time step. Note that a new MDP is calculated at each time step, which is made possible by the high performance of the FastMDP algorithm.

In this pursuit/evasion game, we define a pursuer “capturing" an opponent if it is in a certain region behind the evading aircraft known as the “control zone”. The control zone is normally defined as a cone-like region behind the evading aircraft and describes a region where it is relatively easy to maintain position advantage over the evader and where the evader is vulnerable to weapons employment by the pursuer. In this paper it is more convenient to define it as the position the evader was at 3 seconds previously, which we term here as the “control point”. If the pursuer can closely follow this control point position, then it should also be able to stay within the control zone. If the pursuer is within 100 meters of the control point and relative angle between the two velocity vectors of the aircraft is within 60

degrees, then the pursuer is considered close to the control point. If the pursuer is also pointing at the evader then we consider this a sufficient condition for the pursuer to be able to “capture" the evader (e.g., within range of some weapon). The pursuer must maintain this condition for 30 consecutive time steps in order to successfully “hit" the evader, which is analogous to a weapon taking some time to track the evader. This is indicated visually in the simulation as a red pulsing rectangle around an aircraft that is in danger of being captured.

We build a scoring system that tracks the number of airplanes that have been captured. When a team’s airplane is captured, the opposing team is awarded one point. Thus complete success is when one team reaches a score that equals the number of airplanes on the opposing team. A “win" is described as one team scoring higher than the other, with the other team necessarily incurring a “loss", and a “draw" is when both teams score the same.

We define a metric P_{win} to study the effect of the algorithm over N runs which is defined for a team as the number of wins the team obtained W over the number of runs: $P_{\text{win}} = \frac{W}{N}$. This metric can be applied to 1 vs 1 encounters and can scale to larger teams as well.

The P_{win} measurement alone is not sufficient. Beyond the probability of win, we also wish to define a metric that describes the survivability of the team. In a 10 vs 10 game, it is clearly better when when winning if all 10 of the teammates survive as compared to a win when only 1 of the teammates remain at the end. If we define the number of aircraft at the beginning of the contest as N_{t_0} and the number remaining at the end of the contest as N_{t_f}, then we can define the ratio of teammates that survived a given contest i as $P_{s_i} = N_{t_f}/N_{t_0}$. Over m contests, we define the overall probability of survivibility as $P_s = \frac{1}{m}\sum_{i=1}^{m} P_{s_i}$ where m is the number of contests and is the average probability that the team will survive the contest.

In this paper, our focus is on validating that the algorithm successfully performs pursuit evasion and we do not examine the performance against other methods such as proportional navigation. We also in this paper make no claims of optimality of the approach. We leave the evaluation of the algorithm and a study of optimality for future work and here set out only to show the viability of the approach.

In a pursuit evasion game of this nature, if both teams have equivalent aircraft dynamics and actions, then the natural result is a stalemate which results in a Nash equilibrium where neither team can gain an advantage over the opponent. In order to test the algorithm, we provide the blue team with a slight performance advantage over the red team. If the algorithm is performing pursuit evasion correctly, this should lead to consistent wins by the blue team. If we find that the blue team does not consistently win against the red team, this is a good indicator that there is a flaw in the algorithm.

VI. Results

In Figure 5, results are shown for a typical 1 versus 1 (1v1) encounter. As blue has a performance advantage, it is able to maneuver more effectively and is able to capture the red aircraft. Figure 6 shows the actions selected by the blue aircraft during this run, while Figure 7 shows the values of the 6DOF state variables during the run.

The P_{win} of the blue team for all experiments is shown in Table 4. This is an indicator that the algorithm is functioning correctly as the blue team was given an advantage in the selection of actions and in aircraft dynamics. Better dynamics allow the blue aircraft to maneuver into an offensive position more readily, leading to an expected high P_{win}. Also as expected as the airspace volume becomes more crowded and complex due to the increase in team size, the probability of survivability P_s tends to decrease.

Table 4 Probability of win P_{win} and Probability of survivability P_s of blue team as team size increases over 10 runs

Team Size	P_{win}	P_s
1v1	100%	100%
2v2	100%	100%
3v3	100%	100%
4v4	100%	100%
10v10	100%	99%
100v100	100%	97%

Table 5 Processing time required for each agent on red or blue team as team size increases over 10 runs

Team Size	Mean (ms)
1v1	2.26
2v2	2.50
3v3	2.70
4v4	3.16
10v10	5.55
100v100	27.59

Processing time is shown in Table 5. This shows the average amount of time it takes for an agent to perform one cycle of the FastMDP loop where an MDP is formulated, solved, and the optimal action is selected and applied to the simulation. Processing was performed on a laptop with an Intel i9-8950HK CPU at 2.90 GHz. The code is implemented in Python. This table demonstrates that the algorithm is efficient as the team size grows, though the 100 v 100 case shows that we no longer maintain our target 10 Hz frame rate. Embedded processors with lower throughput should still be able to handle smaller team sizes up to 10 v 10 while maintaining a reasonable frame rate.

Videos of example runs of 1v1, 2v2, 3v3, 4v4, and 10v10 are available for viewing are provided in Table 6. Note that the size of the aircraft is exaggerated by a factor of 3 for improved visibility in the video. See Figure 4 for an image taken from a 10v10 encounter which demonstrates the complexity of the problem and the response of the agents through time as they respond to this complex environment.

Table 6 Links to videos

Team Size	URL
1v1	`https://youtu.be/zGWXxtJUwk8`
2v2	`https://youtu.be/Q9O50cqpVtA`
3v3	`https://youtu.be/6Zok4sj43C4`
4v4	`https://youtu.be/qhI6av3oJN4`
10v10	`https://youtu.be/6twTWNRurwo`

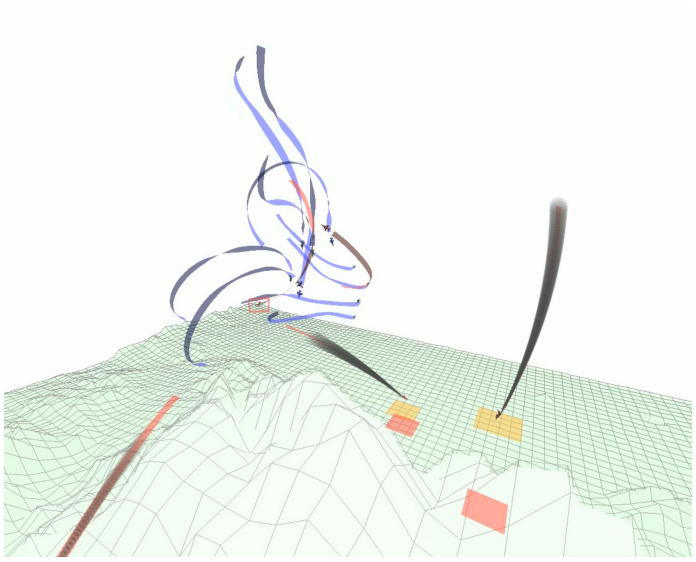
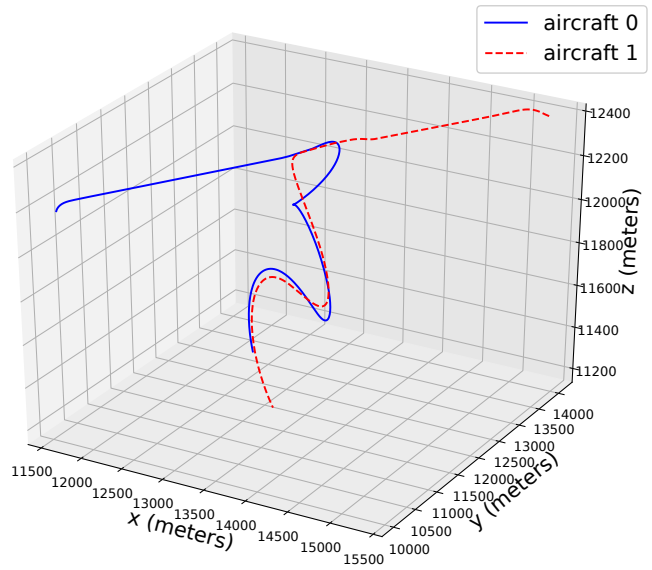
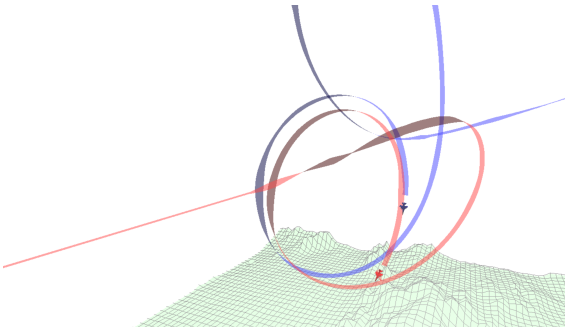


Fig. 4 Screenshot from 10v10 video showing red rectangles indicating an aircraft is in danger of being captured. Once captured, an explosion is indicated, the aircraft loses all thrust, and smoke is emitted by the aircraft until it reaches the ground. As the aircraft approach a minimum safe altitude known as the hard deck (1000 ft above the maximum terrain height) an animated yellow and red square under the aircraft indicate that the aircraft is receiving a penalty for being too close to the ground and is attempting to pull up in response. See Table 6 for links to videos of this and other encounters.



(a) Trajectory of a sample 1v1 pursuit/evasion run



(b) The same 1v1 run in a 3D visualization

Fig. 5 Experimental results showing the performance of the algorithm for a 1v1 pursuit/evasion run. (a) shows the trajectories of two aircraft in a standard Matlab style plot. (b) shows the trajectories in a 3D visualization developed for this paper where ribbons are used to show historical attitude a 3D aircraft is used to more readily show current aircraft attitude. Links to videos are provided for the interested reader in the results sections.

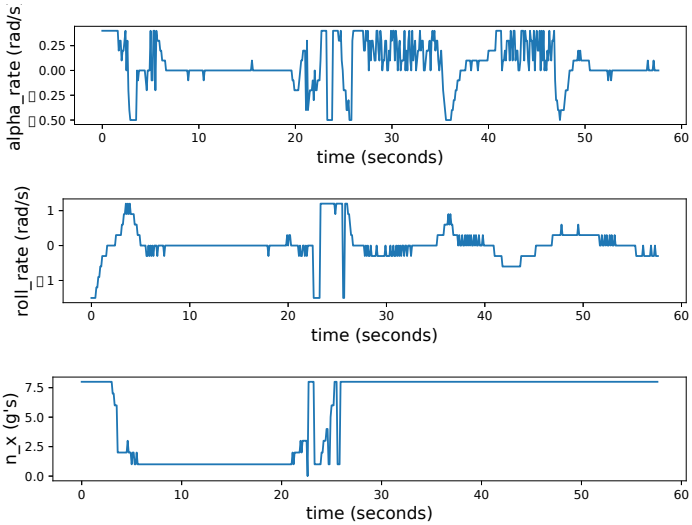


Fig. 6 Experimental results showing the actions taken by the pursuer (blue aircraft) over time. Alpha rate here is analogous to pushing forward or pulling back on the stick. Roll rate is analogous to moving the stick from side to side.

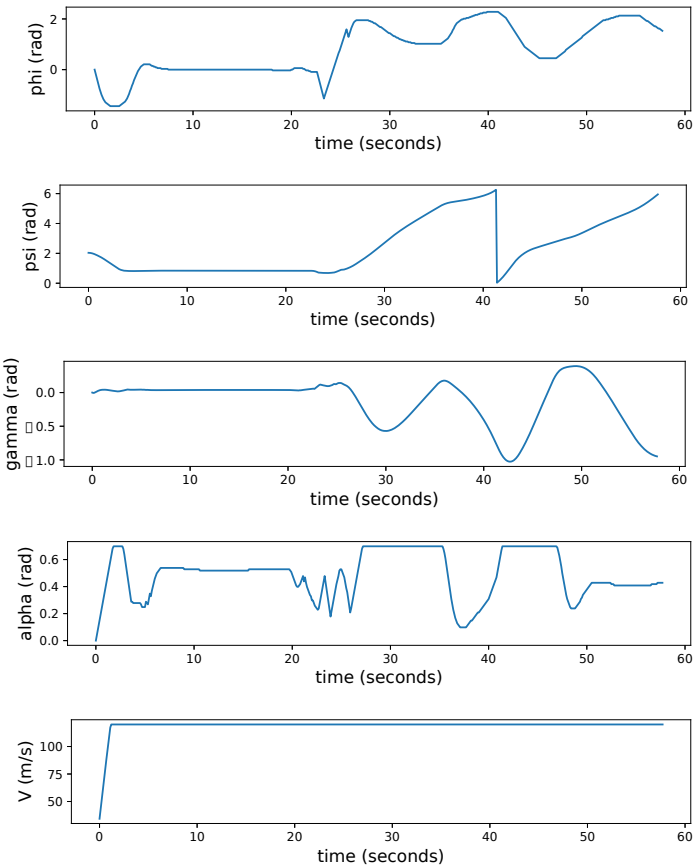


Fig. 7 Experimental results showing the dynamics of the pursuer (blue aircraft) over time.

VII. Conclusion

We have presented an efficient problem formulation for pursuit/evasion problems that scales to large numbers of teams (100v100) while remaining computationally efficient. This method formulates the problem as a Markov Decision Process (MDP), uses a recently proposed approach in [20] to efficiently solve the MDP and is suitable for embedded systems commonly found on aircraft. The use of "risk wells" to represent the potential future actions of friendly and opposing aircraft allows the problem to remain tractable even as the number of aircraft per team increases.

For future work, we plan to explore how to incorporate mutual support, combat tactics, and multiagent cooperation to increase the effectiveness of the teams. We also plan on extending the aircraft model used here to a higher fidelity model to test the algorithm in different areas of the flight envelope. We wish to also compare and contrast this algorithm's performance against other algorithms which can support large scale pursuit/evasion team sizes to compare this algorithm's performance to other approaches.

Also important is to extend this formulation to account for uncertainty. While this is being actively explored, it will require additional theoretical advancements beyond what is presented in this paper.

References

[1] Chief of Naval Air Training (CNATRA), "Flight Training Instruction: BASIC FIGHTER MANEUVERING (BFM) ADVANCED NFO T-45C/VMTS," 2018. URL `https://www.cnatra.navy.mil/local/docs/pat-pubs/P-826.pdf`, accessed: 2019-09-07.

[2] Bertram, J., and Wei, P., "Distributed Computational Guidance for High-Density Urban Air Mobility with Cooperative and Non-Cooperative Collision Avoidance," *AIAA Scitech 2020 Forum*, 2020, p. 1371.

[3] Bertram, J., "A new solution for Markov Decision Processes and its aerospace applications," Master's thesis, Iowa State University, Ames, Iowa, USA, 2020.

[4] Eklund, J. M., Sprinkle, J., and Sastry, S., "Implementing and testing a nonlinear model predictive tracking controller for aerial pursuit/evasion games on a fixed wing aircraft," *Proceedings of the 2005, American Control Conference, 2005.*, IEEE, 2005, pp. 1509–1514.

[5] Schopferer, S., and Pfeifer, T., "Performance-aware flight path planning for unmanned aircraft in uniform wind fields," *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2015, pp. 1138–1147.

[6] Gonçalves, V. M., Pimenta, L. C., Maia, C. A., Dutra, B. C., and Pereira, G. A., "Vector fields for robot navigation along time-varying curves in *n*-dimensions," *IEEE Transactions on Robotics*, Vol. 26, No. 4, 2010, pp. 647–659.

[7] Lawrence, D. A., Frew, E. W., and Pisano, W. J., "Lyapunov vector fields for autonomous unmanned aircraft flight control," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, 2008, pp. 1220–1229.

[8] Guibas, L. J., Latombe, J.-C., LaValle, S. M., Lin, D., and Motwani, R., "Visibility-based pursuit-evasion in a polygonal environment," *Workshop on Algorithms and Data Structures*, Springer, 1997, pp. 17–30.

[9] LaValle, S. M., Lin, D., Guibas, L. J., Latombe, J.-C., and Motwani, R., "Finding an unpredictable target in a workspace with obstacles," *Proceedings of International Conference on Robotics and Automation*, Vol. 1, IEEE, 1997, pp. 737–742.

[10] Kehagias, A., Hollinger, G., and Singh, S., "A graph search algorithm for indoor pursuit/evasion," *Mathematical and Computer Modelling*, Vol. 50, No. 9-10, 2009, pp. 1305–1317.

[11] Lehner, F., "Pursuit evasion on infinite graphs," *Theoretical Computer Science*, Vol. 655, 2016, pp. 30–40.

[12] Shengde, J., Xiangke, W., Xiaoting, J., and Huayong, Z., "A continuous-time Markov decision process based method on pursuit-evasion problem," *IFAC Proceedings Volumes*, Vol. 47, No. 3, 2014, pp. 620–625.

[13] Sun, W., Pan, Y., Lim, J., Theodorou, E. A., and Tsiotras, P., "Min-Max Differential Dynamic Programming: Continuous and Discrete Time Formulations," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 12, 2018, pp. 2568–2580.

[14] Bellman, R. E., "Dynamic Programming," 1957.

[15] McGrew, J. S., How, J. P., Williams, B., and Roy, N., "Air-combat strategy using approximate dynamic programming," *Journal of guidance, control, and dynamics*, Vol. 33, No. 5, 2010, pp. 1641–1654.

[16] Carr, R. W., "Optimal Control Methods for Missile Evasion," Tech. rep., AIR FORCE INSTITUTE OF TECHNOLOGY WRIGHT-PATTERSON AFB OH WRIGHT-PATTERSON, 2017.

[17] Park, H., Lee, B.-Y., Tahk, M.-J., and Yoo, D.-W., "Differential game based air combat maneuver generation using scoring function matrix," *International Journal of Aeronautical and Space Sciences*, Vol. 17, No. 2, 2016, pp. 204–213.

[18] Zhang, X., Liu, G., Yang, C., and Wu, J., "Research on Air Confrontation Maneuver Decision-Making Method Based on Reinforcement Learning," *Electronics*, Vol. 7, No. 11, 2018, p. 279.

[19] Sutton, R. S., and Barto, A. G., *Reinforcement learning: An introduction*, Vol. 1, MIT press Cambridge, 1998.

[20] Bertram, J. R., Yang, X., Brittain, M., and Wei, P., "Online Flight Planner with Dynamic Obstacles for Urban Air Mobility," *2019 Aviation Technology, Integration, and Operations Conference*, 2019.

[21] Huynh, H., Costes, P., and Aumasson, C., "Numerical optimization of air combat maneuvers," *Guidance, Navigation and Control Conference*, 1987, p. 2392.