A 3D game theoretical framework for the evaluation of unmanned aircraft systems airspace integration concepts

Berat Mert Albaba, Negin Musavi, Yildiray Yildiz

Bilkent University, 06800 Bilkent, Ankara, Turkey
University of Illinois at Urbana Champaign, 306 Engineering Hall, MC 266, 1308 West Green Street, Urbana, IL 61801, USA

ARTICLE INFO

Keywords: UAS integration into NAS, Reinforcement Learning, Behavioral Game Theory, Human Modeling

ABSTRACT

Predicting the outcomes of integrating Unmanned Aerial System (UAS) into the National Airspace System (NAS) is a complex problem, which is required to be addressed by simulation studies before allowing the routine access of UAS into the NAS. This paper focuses on providing a 3-dimensional (3D) simulation framework using a game-theoretical methodology to evaluate integration concepts using scenarios where manned and unmanned air vehicles co-exist. In the proposed method, the human pilot interactive decision-making process is incorporated into airspace models which can fill the gap in the literature where the pilot behavior is generally assumed to be known a priori. The proposed human pilot behavior is modeled using a dynamic level-k reasoning concept and approximate reinforcement learning. The level-k reasoning concept is a notion in game theory and is based on the assumption that humans have various levels of decision making. In the conventional “static” approach, each agent makes assumptions about his or her opponents and chooses his or her actions accordingly. On the other hand, in the dynamic level-k reasoning, agents can update their beliefs about their opponents and revise their level-k rule. In this study, Neural Fitted Q Iteration, which is an approximate reinforcement learning method, is used to model time-extended decisions of pilots with 3D maneuvers. An analysis of UAS integration is conducted using an Example 3D scenario in the presence of manned aircraft and fully autonomous UAS equipped with sense and avoid algorithms.

1. Introduction

Although unmanned aircraft systems (UASs) have operational and cost advantages over manned aircraft in many applications, they do not have routine access to the National Airspace (NAS). The aviation industry, being very sensitive to safety, needs strong evidence that the UAS integration will not have any negative impact on the existing airspace system in terms of safety (Huerta, 19:2013; Group, 2013) before they are granted routine access into the NAS. Examples of risk analyses and safety assessments of UAS integration into NAS can be observed in the studies conducted by Zhang et al. (2018), Ferreira et al. (2018). Until technologies, standards, and procedures for the safe integration of UAS into the airspace are matured, there will not be enough data accumulated about the issue, and it will be hard to predict the effectiveness of the related technologies and concepts. Although research efforts exist to develop a safe and efficient real test environment for UAS integration, flight tests are expensive and experimental failures can cause severe economic loss. Therefore, employing simulations is currently the most efficient way to understand the effects of UAS integration on the air traffic.

* Corresponding author.
E-mail addresses: mert.albaba@bilkent.edu.tr (B.M. Albaba), nmusavi2@illinois.edu (N. Musavi), yyildiz@bilkent.edu.tr (Y. Yildiz).

https://doi.org/10.1016/j.trc.2021.103417
Received 18 July 2021; Accepted 27 September 2021
Available online 23 October 2021
0968-090X/© 2021 Elsevier Ltd. All rights reserved.
system (DeGarmo, 2004). Although simulation frameworks consisting of multiple UASs are proposed by Zhao et al. (2019) and Al-Mousa et al. (2019), simulation studies need to be conducted with hybrid airspace system (HAS) models, where manned and unmanned vehicles coexist.

HAS models in the literature are generally based on the assumption that the pilots of manned aircraft always behave as expected, without deviating from ideal behavior (Maki et al., 2012; Pérez Batlle et al., 2012; Martel et al., 2010; Kochenderfer et al., 2008; Billingsley, 2006; Kuchar et al., 2004). Most of the existing HAS models are designed to evaluate and test the performance of collision avoidance systems in single encounter scenarios in which the intruder (generally a manned aircraft) has a pre-defined behavior with no consideration of the decision making process of the pilot. These models are valuable and essential at the initial stages of evaluating a new method but it is not realistic to expect that the pilot, as a decision-maker, will always behave deterministically and in a pre-defined manner. It is not always predictable, for example, how pilots will respond to the traffic control alert system (TCAS) (Wiener and Nagel, 1988). TCAS is an onboard collision avoidance system, which observes and tracks surrounding air traffic, detects conflicts, and suggests avoidance maneuvers to the pilots. It is shown that only 13% of pilot responses match the deterministic pilot model that was assumed for TCAS development (Lee and Wolpert, 2012; Kuchar and Drumm, 2007). Therefore, incorporating human decision-making processes in HAS models has a strong potential to improve the predictive power of these models.

In prior works (Musavi et al., 2017; Musavi et al., 2016), authors have created HAS models with human decision-making models, inspired by a game-theoretical methodology known as semi-network-form games (Lee and Wolpert, 2012), where the pilot behavior was not assumed to be known a priori but obtained using 1) the level-k reasoning concept which is a game-theoretical approach used to model multiple strategic player interactions, where it is assumed that humans have various levels of reasoning, level-0 being the lowest level, and 2) reinforcement learning, which helps model time extended decisions as opposed to assuming one-shot decision making. Although these studies introduced one of the very first examples of HAS models where several decision-makers can be modeled simultaneously in a time-extended manner, they had two limitations: First, HAS models were developed for a 2-dimensional (2D) airspace. Second, the policies, maps from observation spaces to action spaces, obtained for the decision-makers remain unchanged during their interaction. In the proposed framework, these limitations are removed, and a 3D HAS model is introduced where the strategic decision-makers can modify their policies during interactions with each other. Therefore, compared to Musavi et al. (2017) and Musavi et al. (2016), a much larger class of interactions can be modeled.

It is shown in the literature that 1) in repeated strategic interactions, where agents consider other agents’ possible actions before determining their own, agents with different cognitive abilities change their behavior during the interaction (Gill and Prowse, 2016) and 2) there is a positive relationship between cognitive ability and reasoning levels (Gill and Prowse, 2016; Gill and and Prowse, 2012). These observations lead to agents with different levels of reasoning who can observe their opponents’ behavior during repeated interactions, update their beliefs on their opponents’ reasoning level, and change their own level-k rule against them. In studies presented by Gill and Prowse (2012) and Gill and Prowse (2016), a systematic level-k structure is introduced where players can update their beliefs about their opponents, and switch their own level rule up one level during their interactions. There are also other level-k rule learning models in the literature such as the ones presented by Chong et al. (2016) and Ho and Su (2013), where the agent levels can reach up to infinity. This is not a problem for the applications investigated by Chong et al. (2016) and Ho and Su (2013), in which obtaining level-k rules ($k = 0,1,2,\ldots,\infty$) are straight forward and has an analytical solution. Since it is computationally expensive to obtain higher levels, and in certain experimental studies it is shown that humans, in general, have a maximum reasoning level of 2 (Costa-Gomes et al., 2009), the existing level-k rule learning methods may not be suitable for the application considered in this work where 200 decision-makers are modeled simultaneously in a time extended manner. Here, we propose a simpler method for modeling level-k rule updates during interactions by a) limiting the levels up to 2 and b) allowing rule updates only if a trajectory conflict is predicted.

Different from the 2D HAS models developed by Albaba and Yildiz (2019), Musavi et al. (2017, 2016), in this study, the game-theoretical modeling framework is developed for a 3D HAS model, which allows covering a much larger class of integration scenarios. The reinforcement learning algorithm used in the authors’ earlier works (Albaba et al., 2019; Li et al., 2017; Yildiz et al., 2014; Backhaus et al., 2013) employ tables to store the Q-values of all state (location of the intruder, approach angle of the intruder, best trajectory action, best destination action, and previous action)-action (turn left, turn right, go straight) pairs, which define how preferable it is to take a certain action given the observations/states. This poses a challenge for the application of the method to systems with large numbers of state-action pairs, such as the proposed 3D HAS model in this study. To circumvent this issue, Neural Fitted Q-Learning (NFQ) method (Gabel et al., 2011; Riedmiller et al., 2007; Riedmiller, 2005), an approximate reinforcement learning algorithm, is utilized. Approximate reinforcement learning methods use function approximators to represent the Q-value function (van Otterlo, 2012). In other words, instead of saving Q-values for each state-action pair, the Q-value function is approximated by a function approximator. In the case of NFQ, a neural network is used as the function approximator. NFQ approach also allows using a continuous observation space, which also contributes to obtaining a more precise definition of the agents’ observations, compared to conventional approaches, where a discretized observation space is required.

In the simulations, pilot models that are obtained using the proposed game theoretical modeling framework are used in complex scenarios, where UAS and manned aircraft co-exist, to analyze the probable outcomes of HAS interactions. HAS scenarios contain
interacting humans (pilots) who also interact with multiple UASs with their own sense and avoid (SAA) systems. It is noted that automation algorithms other than SAA systems, such as TCAS, and possible air traffic management instructions can also be incorporated into the proposed framework. During the simulations, UASs fly autonomously based on pre-programmed flight plans but they can deviate from their plans to resolve a possible conflict with the help of their SAA algorithms. In these simulations, as an example to demonstrate how the proposed framework can be utilized, the effect of responsibility assignment for conflict resolutions on the safety and performance of the HAS is analyzed (see Johnson et al. (2012) for the importance of these variables and responsibility assignment for UAS integration). Furthermore, the safety and performance of UASs are investigated in different scenario settings. It is noted that the scope of this research is not limited to the type of scenarios used for demonstration purposes. The proposed modeling framework is flexible enough to test a diverse set of flight conditions, including free flight or a more centralized and automated system. The proposed pilot models are validated using MIT Lincoln Laboratory’s data-validated aircraft encounter model for NAS.

The organization of the paper is as follows: In Section 2, the HAS scenario for UAS integration into the NAS is described in detail. In Section 3, the proposed pilot decision modeling method is explained. In Section 4, validation results are provided. In Section 5, simulation results are given. Finally, conclusions are given in Section 6.

2. UAS Integration Scenario

In order to evaluate the possible outcomes of integrating Unmanned Aircraft System (UAS) into the National Air Space (NAS), a Hybrid Air Space (HAS) scenario, where manned and unmanned aircraft co-exist, is designed and explained in this section.

2.1. Simulation Framework

The simulation framework consists an airspace which has the size of 196850.39ft (length) × 984251.97ft (width) × 50000ft (height) (60000m × 300000m × 15240m). The manned aircraft in the framework execute maneuvers based on the pilot model obtained using a combination of reinforcement learning and level-k reasoning, the details of which are explained in Section 3. 200 manned aircraft are placed randomly, with random position and velocity initializations, in the airspace. Furthermore, multiple UASs are located in specified points in the airspace and move based on their pre-programmed flight plan from one waypoint to another. Fig. 1 shows a snapshot of the simulated airspace with multiple manned aircraft and three UASs moving through their multiple waypoints. All aircraft whether manned or unmanned are flying at different altitudes, and this snapshot depicts a 2D projection of their configuration, on the horizontal plane. All aircraft, manned or unmanned, have continuous dynamics, which are explained in the following sections.

Fig. 1. A 2D projection of the simulated airspace with multiple manned aircraft (red) and three UASs (cyan). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
2.2. UAS Conflict Detection and Avoidance Logic

UASs are assumed to have the dynamics of RQ-4 Global Hawk with an operation speed of 340 knots (Dalamagkidis et al., 2011). UASs are also equipped with sense and avoid (SAA) systems, which enable them to detect trajectory conflicts and to initiate evasive maneuvers, if necessary. If no conflict is detected, UASs continue to follow their mission plan. Either receiving a conflict resolution command from the SAA system or flying based on their pre-defined flight plan, UASs always receive a velocity command during the flight. The UASs velocity vector variation is modeled as first-order dynamics with a time constant of 1s (Mujumdar and Padhi, 2011) which is represented as

\[
\ddot{\mathbf{v}} = -(\mathbf{v} - \mathbf{v}_d),
\]

where \(\mathbf{v}\) and \(\mathbf{v}_d\) are the current and the desired/commanded velocity vectors, respectively. The two SAA logics that are utilized in this study are developed by Mujumdar and Padhi (2011), which is referred to as SAA1, and Fasano et al. (Fasano et al., 2008), which refers to as SAA2. Both of the SAA logics contain two phases; a conflict detection phase and a conflict resolution phase. The conflict detection phase is the same for both SAA1 and SAA2. A conflict is detected if the minimum distance between the UAS and the intruder aircraft is calculated to be less than a minimum required distance, \(R\), during a predefined time interval. The minimum distance is calculated by projecting the trajectories of the UAS and the intruder aircraft in time. Once the conflict is detected, SAA1 and SAA2 suggest their own velocity adjustment commands to resolve the conflict. The velocity adjustment command of the SAA1 and SAA2 logics, \(\mathbf{v}^{A1}_{d1}\) and \(\mathbf{v}^{A2}_{d2}\), are given as

\[
\mathbf{v}^{A1}_{d1} = \frac{-\mathbf{v}_A \left( \mathbf{v}_A \mathbf{v}_{AB} \right)}{\|\mathbf{v}_A\|} \left( R - \|\mathbf{v}_m\| \right) \frac{\mathbf{v}_m}{\|\mathbf{v}_m\|},
\]

\[
\mathbf{v}^{A2}_{d2} = \left[ \frac{\nu_{AB} \cos(\eta - \zeta)}{\sin(\zeta)} \left( \sin(\zeta) \frac{\mathbf{v}_{AB}}{\|\mathbf{v}_{AB}\|} - \sin(\eta - \zeta) \frac{\mathbf{r}}{\|\mathbf{r}\|} \right) \right] + \mathbf{v}^B,
\]

where \(\mathbf{v}_A\) is the velocity vector of the UAS, \(\mathbf{v}_B\) is the velocity vector of the intruder, \(\mathbf{r}\) is the relative position between the UAS and the intruder, \(\mathbf{v}_{AB}\) is the relative velocity between the UAS and the intruder, \(\zeta\) is the angle between \(\mathbf{r}\) and \(\mathbf{v}_{AB}\), and \(\eta\) is calculated as

\[
\eta = \sin^{-1} \left( \frac{R}{\|\mathbf{r}\|} \right) \mathbf{r}_0.
\]

\(\mathbf{r}_0\) refers to the initial relative position vector between the UAS and the intruder. If multiple conflicts are detected, UAS starts an evasive maneuver to resolve the conflict that is predicted to happen earliest. The velocity adjustment suggested by the SAA2 logic guarantees minimum deviation from the trajectory. On the other hand, with the SAA1 logic, UAS moves to resolve the conflict until it retains the minimum safe distance with the intruder. Further examples of SAA development studies for a safe integration of UAS into the airspace can be seen in the studies of Johnson and Shively (2018), Meer et al. (2019), D’Amato et al. (2018) and Allignol et al. (2016).

2.3. Manned Aircraft Dynamics

All manned aircraft are assumed to be in their en-route phase of travel with constant speed, \(v\), in the range of [170–390] nm/h (knots). Once the pilot gives a heading or pitch command, the aircraft moves to the desired heading and pitch, \(\psi_d\) and \(\theta_d\), in the constant speed mode, where the heading and pitch change is modeled with the standard rate turn: a turn in which an aircraft changes its heading at a rate of 3° per second (360° in 2 min) (Federal Aviation Administration, 2009). This is approximated as a first-order dynamics with a time constant of 10s (45(1 − 1/3) ≈ 10). Therefore, the aircraft heading and pitch angle dynamics can be given as

\[
\dot{\psi} = -\frac{1}{10}(\psi - \psi_d),
\]

\[
\dot{\theta} = -\frac{1}{10}(\theta - \theta_d).
\]
The velocity, $\vec{v} = (v_x, v_y, v_z)$, is then obtained as

$$v_x = \|\vec{v}\| \sin(\psi) \cos(\theta),$$  \hspace{1cm} (7)

$$v_y = \|\vec{v}\| \cos(\psi) \cos(\theta),$$  \hspace{1cm} (8)

$$v_z = \|\vec{v}\| \sin(\theta).$$ \hspace{1cm} (9)

### 2.4. Observation Space

Automatic Dependent Surveillance-Broadcast (ADS-B) provides positions and velocities of the aircraft in the environment. However, during the decision-making process, a pilot cannot process and utilize all this information. Thus, in this work, it is assumed that the pilots can process, or observe, the information from a limited portion of the nearby airspace, which allows to model pilot limitations, including the limitations at visual acuity and perception depth, as well as the limited viewing range of an aircraft. The observation space is considered as a sphere centered at the location of the pilot. This space is depicted in Fig. 2, where horizontal and vertical cross-sections are separately drawn. Since the standard separation for manned aviation is 3–5 nautical miles (nm) (Pérez Batlle et al., 2012), the radius of the observation space, $r$, is taken as 5 nm. When an intruder aircraft moves toward the observation space, the approach geometry is defined by two angles: $\phi_H$, in the horizontal plane, and $\psi_V$, in the vertical plane. Aircraft’s angular orientation with respect to his/her ideal trajectory is also defined by two angles: $\beta_H$, in the horizontal plane, and $\phi_V$, in the vertical plane. In Fig. 2, the intruder is moving toward the observation space of the ego agent with $\phi_H = -45^\circ$ and $\psi_V = +45^\circ$, and the ego agent’s orientation with respect to the ideal path is such that $\beta_H = -60^\circ$ and $\beta_V = -90^\circ$. Aircraft orientations with respect to each other are also included in the observation space and coded as different **encounter types**. Fig. 3 depicts these encounter geometries projected in the horizontal plane. There are 8 types of encounters, which are represented as $C_i$, $i = 1$,$\ldots$,$8$. The first four encounters are defined in the horizontal plane and the last four in the vertical plane. Also, the observation space contains the current action, the previous action, and the difference between the initial vertical position and the current vertical position.

The observation vector, which is based on the observation space and used to train pilot policies, at time-step $t$ is represented as $O_t = [\text{sign} (\beta_H), \text{sign} (\psi_V), \text{sign} (\phi_H), \text{sign} (\phi_V), \text{intruder status}, \text{encounter type}, \text{action}_{t-1}, \text{action}_t, \text{vertical difference}]^T$, where $\text{sign}(x)$ takes the value $+1$ if $x > 0$, $-1$ if $x < 0$ and 0 otherwise. Whenever an intruder is detected in the observation space, **intruder status** takes the value of 1, and 0 otherwise. Encounters with other aircraft are represented by the **encounter type**, which is a 2-length vector, whose first element is for the horizontal plane and the second one is for the vertical plane. The first element of **encounter type** takes the values of $-1$, $-0.5$, 0.5, 1, corresponding to $C_1$, $C_2$, $C_3$ and $C_4$, respectively, and 0 otherwise. Second element of the **encounter type** takes the values of $-1$, $-0.5$, 0.5, 1 for $C_5$, $C_6$, $C_7$ and $C_8$, and 0 otherwise. The previous and the current action of the pilot are included in the observation space as $\text{action}_{t-1}$ and $\text{action}_t$. Finally, **vertical difference** $= (z_{\text{initial}} - z_t)/s$ encodes the normalized vertical difference from the start, where $z$ is the vertical position and $s$ is the speed of the aircraft.

![(Fig. 2. Observation space of a pilot, which is a sphere centered on the location of the aircraft.)](image-url)
2.5. Action Space

Continuous maneuvers of the aircraft are created through a two-step process. In the first step, there exist discrete high-level pilot decisions, which are called the actions. These can be considered as approximations of continuous decision variables. In the second step, these decisions are filtered through aircraft dynamics, explained in Section 2.3, to create continuous aircraft movement. Discrete actions consist of $\pm 45^\circ$ in the horizontal plane and $\pm 10^\circ$ in the vertical plane. It is assumed that the flight control system successfully realizes these actions with transient response dynamics resulting in a standard rate turn motion for aircraft explained in Section 2.3.

3. Methods for Modeling Pilot Decisions

For the modeling of the decision-making process of pilots, two methodologies are employed: dynamic level-k reasoning and neural fitted Q iteration.

3.1. Neural Fitted Q-iteration

Reinforcement learning is a mathematical learning approach that builds on rewards and punishments (van Otterlo, 2012). The learning agent interacts with the environment through taking actions according to its observations and receives a reward or punishment as a result. In detail, at each time-step $t$, agent makes an observation of its state $s_t$ from the state space $S$ and takes an action $a_t$ from the action space $A$. Then, agent transitions to the next state $s_{t+1}$ by taking a reward/punishment $r_t$. State space contains all possible observations of the agents and similarly, action space covers all possible actions in the environment. Agents behaviors/decisions are defined by its policy, $\pi$, which is a mapping from state space to action space ($S \rightarrow A$) and contains the action preferences of the agent for each state. The main purpose of the agent is maximizing the cumulative discounted future rewards. In other words, agents tries to find the optimal policy

$$\pi^* = \arg\max_{\pi} \mathbb{E} \left( \sum_{t} \gamma^t r_t \right),$$

where $\gamma$ is a discount factor. For a clear definition of the optimal policy, first, the value function, $V^\pi(s_t)$, that expresses the expected discounted future cumulative reward when the agent starts from state $s_t$ and follows the policy $\pi$ is defined as

**Algorithm 1.** Neural Fitted Q-Iteration

1: Initialize the experience memory $D$
2: Initialize the network $Q$ by initializing weights with uniform random initialization, by sampling a value between $(-1, 1)$ for each weight term.
3: for episode = 1 to $M$ do

(continued on next page)
Similarly, the action-value function, $Q$, estimates the value of taking an action $a_t \in \Lambda$ in state $s_t$ in terms of maximizing the future reward. The action-value function and its optimal value are defined as

$$Q^*(s_t, a_t) = E[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | s_t, a_t].$$

$$Q^*(s_t, a_t) = \max_a Q^*(s_t, a).$$

The relation between the optimal value and Q-functions is expressed as

$$V^*(s_t) = \max_a Q^*(s_t, a).$$

Finally, the optimal policy in state $s_t$ is defined as

$$\pi^*(s_t) = \arg \max_a Q^*(s_t, a).$$

Through the training, the Q-function should be updated iteratively with the experience of the agent in order to reach the optimal policy at the end. In classical reinforcement learning, i.e. Q-learning, Q-function is updated as

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a)),$$

where $t$ is the time-step, $\alpha$ is the learning rate, and $\gamma$ is the discount factor. In neural fitted Q-iteration (NFQ), instead of directly calculating Q-values, they are approximated by a neural network, whose inputs are observed state-action pairs. In order to approximate Q-values properly, the neural network tries to minimize

$$L = (Q(s_t, a_t) - (r_{t+1} + \gamma \max_a Q(s_{t+1}, a)))^2$$

which is the deviation of the approximated Q-value, $Q(s_t, a_t)$, from the target value, $r_{t+1} + \gamma \max_a Q(s_{t+1}, a)$. The algorithm of the NFQ is presented in Algorithm 1. In this method, during an episode, agent collects experiences as quadruples, $(s_t, a_t, r_t, s_{t+1})$, and at the end of the episode, over all these experiences, neural network weights are updated by using the loss defined in (18) with resilient back-propagation (Riedmiller and Braun, 1993). In this work, in order to allow the agent to explore state-action pairs, the epsilon-greedy algorithm with a dynamic exploration rate is used as

$$\epsilon = \max(0.1, 0.99977^k)$$

where $k$ is the episode number. Thus, the exploration rate starts at 1 and decreases to 0.1 in 10000 episodes.

### Algorithm 1: NFQ

4: for $t = 1$ to terminal do
5: Observe the state $s_t$ and take an action $a_t$ with epsilon greedy, i.e. choose a random action with probability of $\epsilon = \max(0.1, (0.99977)^{\text{episode}})$, which allows to take more random actions initially but less later during the training.
6: Execute action $a_t$ and observe the reward $r_t$ and the transitioned state $s_{t+1}$
7: Store the experience $(s_t, a_t, r_t, s_{t+1})$ in $D$
8: if $s_t$ is terminal, i.e. agent crashes or finishes, then
9: break
10: end if
11: end for
12: if size($D$) $>$ 0 then
13: Set matrices $y$ and $x$ as empty.
14: for $j = 1$ to size($D$) do
15: Set $x_j = (s_t, a_t)$
16: Set $y_j = r_t + \gamma \max_a Q(s_{t+1}, a; W)$
17: end for
18: Perform resilient back-propagation by using $\text{input} = x, \text{output} = y$ and update the weight matrix $W$ using the cost function $|Q(x; W) - y|^2$.
19: end if
20: end for

$$V^*(s_t) = E[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | s_t].$$ (11)

The optimal value function, therefore, is given as

$$V^*(s_t) = \max_a V^*(s_t).$$ (12)

Similarly, the action-value function, $Q^*$, estimates the value of taking an action $a_t \in \Lambda$ in state $s_t$ in terms of maximizing the future reward. The action-value function and its optimal value are defined as

$$Q^*(s_t, a_t) = E[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | s_t, a_t].$$ (13)

$$Q^*(s_t, a_t) = \max_a Q^*(s_t, a).$$ (14)

The relation between the optimal value and Q-functions is expressed as

$$V^*(s_t) = \max_a Q^*(s_t, a).$$ (15)

Finally, the optimal policy in state $s_t$ is defined as

$$\pi^*(s_t) = \arg \max_a Q^*(s_t, a).$$ (16)
Remark 1. Other approximate reinforcement learning algorithms, such as Deep Q-Learning (DQN) (Mnih et al., 2013) or Double Deep Q-Learning (DDQN) (Van Hasselt et al., 2015), can also be used to accommodate the exponentially growing state numbers due to the 3D geometry. In this work, we preferred to use the NFQ algorithm due to its relative simplicity and faster training time (Duarte et al., 2019).

3.1.1. Reward Function

Since the goal of the reinforcement learning algorithm is to maximize future discounted cumulative reward, the reward function is the utility or the happiness function and represents the goals/preferences of the agent. In this work, the reward function is defined as

\[ r = w_1 C + w_2 S + w_3 A + w_4 P + w_5 D, \]

where \( C \) is the number of aircraft within the collision radius (152.4 m in the horizontal direction and 100 ft in the vertical direction (Planning and Office, 2007)), \( S \) is the number of air vehicles within the separation region (5 nm in the horizontal direction (Pérez Batlle et al., 2012) and 1000 ft in the vertical direction based on the Reduced vertical separation minima (Planning and Office, 2007)), \( A \) represents whether the aircraft is getting closer to the intruder or going away from the intruder and takes the values of 1, for getting closer, or 0, for going away, \( P \) represents whether the aircraft gets closer to or goes away from its trajectory vector and takes the values of 0, for getting closer, or 1, for going away, and \( D \) represents the normalized altitude change and takes the value of \(|z - z_0|/s\), where \( z \) is the current vertical position, \( z_0 \) is the initial vertical position and \( s \) is the speed of the aircraft. \( w_i \) are the weights, representing the relative importance of each term.

3.2. Dynamic Level-k Reasoning

Level-k reasoning is a game-theoretical model where the main idea is that humans have various levels of reasoning in their decision-making process (Chong et al., 2016). It has been observed that the reasoning levels are related to the cognitive abilities of humans (Gill and Prowse, 2016). The level hierarchy is iteratively defined such that the level-k rule is the best response to the level-(k-1) rule. A level-1 decision maker (DM), for example, assumes that the other agents in the scenario are level-0 and takes actions accordingly to provide the best response. A level-2 DM takes actions to give the best response to other DMs that have level-1 reasoning and so on. To clarify, for player \( i \), level-1 and level-2 strategies, \( \rho_1 \) and \( \rho_2 \), are defined as

\[ \rho_1 = \arg\max_{p \in P} u_i(p | \rho_0) \]
\[ \rho_2 = \arg\max_{p \in P} u_i(p | \rho_1) \]

where \( P \) contains all possible strategies, \( u_i(a|b) \) defines the utility function of player \( i \) when player \( i \) follows the strategy \( a \) and the other players have strategy \( b \). Level-0 decision-making strategy, i.e. level-0 DM, does not consider other players while taking actions. Hence, a level-0 rule represents a nonstrategic DM which can also be considered as reflexive since it only reacts to the immediate observations. In this study, a level-0 pilot flies an aircraft with constant heading and pitch angles starting from its initial position toward its destination.

In its conventional form described above, level-k reasoning helps to model the interactions between the DMs where a level-k DM assumes that the other DMs have level-(k-1) reasoning. Although this approach proved to be successful in modeling short-term or one-shot interactions, it misses the point that agents, during their interactions, may update their assumptions about the other agents and in turn update their own behavior. To remedy this problem, we introduce a closed-loop strategy, which allows the agents to dynamically update their reasoning levels if a trajectory conflict is detected. For a two-player game, this strategy is presented as

\[ \rho^1 = \begin{cases} \arg\max_{p \in P} u_1(p | \rho_1), & \text{if} (c_i + c_{i-1} = 2) \text{and} ((k - l \geq 2) \text{or} (X \geq 0) \text{and} (k = l)) \\ \rho_1 \end{cases}, \]
\[ \rho^2 = \begin{cases} \arg\max_{p \in P} u_2(p | \rho_2), & \text{if} (c_i + c_{i-1} = 2) \text{and} ((l - k \geq 2) \text{or} (X < 0) \text{and} (k = l)) \\ \rho_1 \end{cases}, \]

where \( \rho^1 \) and \( \rho^2 \) are the strategies of player 1 and player 2, \( k \) and \( l \) are the levels of player 1 and player 2 before the encounter, \( X \) is a random variable with density function \( f_X(x) = (e^{-(x^2/2)})/(\sqrt{2\pi}) \), which corresponds to the standard normal distribution, and \( c_i \) is the conflict variable that takes the value of 1 if there is a conflict in time-step \( t \), and 0 otherwise.

3.3. Combination of Neural Fitted Q-Iteration and Level-k Reasoning

For the generation of strategic decision-makers that can make time-extended decisions, level-k reasoning is combined with neural fitted q-iteration. As stated previously, level-0 is a non-strategic decision-maker that follows predefined rules. In order to obtain the level-1 policy, first of all, the training environment is populated with level-0 agents. Then, a learning agent, i.e. ego agent, with no prior knowledge, is placed in this environment, and the training is started. With reinforcement learning, the ego agent learns the actions that are the best responses to level-0 agents in each state. Once the training ends, the ego agent becomes a level-1 agent with the level-1 policy, \( \pi^1 \). Higher reasoning level decision-makers are obtained using the same process. A general framework to obtain a level-k policy is explained briefly in Algorithm 2.
Algorithm 2. Combination of Level-k and Neural Fitted Q-Iteration

1: Load the previously obtained level-(k-1) policy, $\pi^{k-1}$.
2: Place agents to the environment and assign level-k-1 policy as the strategy of all the agents in the environment, $x_i = \pi^{k-1}$ for $i = 1, 2, \ldots, N$, where $N$ is the number of non-learning agents in the environment.
3: Assign an empty (randomly initialized) policy to the ego agent, $\pi^\text{ego} = \pi^\text{empty}$.
4: After placing the ego agent in the created environment, start the reinforcement learning training, which is explained in Algorithm 1.
5: At the end of the training, ego agent learns the policy containing best responses to $\pi^{k-1}$. Save the resultant policy as level-k policy $\pi_k$.

In this work, the total number of agents in the environment, $N$, is 200.

Remark 2. In the proposed approach, each level-k agent is trained by assigning level-(k-1) behavior for the rest of the players. This level-(k-1) behavior is either obtained in earlier training phases or predefined if it corresponds to level-0. Therefore, the computational complexity of the proposed method is the same as that of the NFQ method (Riedmiller, 2005).

4. Validation Results

For the validation of the proposed 3D airspace model, MIT Lincoln Laboratory’s data-validated aircraft encounter model presented by Kochenderfer et al. (2008) is employed. 50 different encounter scenarios generated by the Lincoln Lab’s model are publicly available as text files cor_ac1.txt and cor_ac2.txt, which include positions, speeds, accelerations, and heading angles of the aircraft for the duration of 50s. For each scenario, the same initial conditions used in the Lincoln Lab’s model are assigned to the pilot models obtained by the proposed method, and the resulting trajectories are compared. In order to evaluate the similarity between two encounter models, two different distance metrics are used: 1. Fréchet Distance, and 2. Euclidean Distance.

4.1. Fréchet Distance

Fréchet distance is a trajectory similarity metric (Toohey and Duckham, 2015). It measures the similarity by considering both time-stamps and locations of trajectory points. Since the time-stamps of the points are considered, the similarity between trajectories with different lengths can be measured. Fréchet distance can be described with one example: In a scenario where a person is walking a dog on a leash, the person and the dog may follow different trajectories by varying their speed but cannot go backward, Fréchet distance measures the shortest leash length that is enough to traverse both curves (Eiter and Mannila, 1994), i.e. that allows the person and the dog to follow their trajectory. Formally, as Fréchet distance between two trajectories, $t_1 : [a, b] \rightarrow V$ and $t_2 : [a’, b’] \rightarrow V$ can be calculated as

$$\delta_F(t_1, t_2) = \inf_{\alpha, \beta} \max_{a’ \leq \alpha(t) \leq b’, a \leq \beta(t) \leq b} d(\alpha(t), \beta(t)).$$  (25)

where $(V, d)$ is a metric space, $\alpha$ and $\beta$ are continuous non-decreasing functions (Eiter and Mannila, 1994). In this work, discrete version of the Fréchet distance is used (Eiter and Mannila, 1994). As distance function, $d$, Euclidean distance is used.

4.2. Euclidean Distance

Euclidean distance computes the mean of the square distance between points of the given trajectory pairs. However, Euclidean distance does not consider the time information, and given trajectories must be at the same length. In this work, the longer trajectory is cropped to the length of the shorter, and the Euclidean distance is calculated over trajectory points with the same time-stamp. Mean Euclidean distance is calculated as

$$\delta_E(t_1, t_2) = \sqrt{\sum_i \frac{(t_1(i) - t_2(i))^2}{|t_1 - t_2|}}.$$  (26)

where $t_1$ and $t_2$ are the trajectories and $i$ represents the time-stamp.

4.3. Comparison Results

We compared the trajectories of 50 different encounter scenarios, presented in cor_ac1.txt and cor_ac2.txt, with the trajectories created by our game theoretical (GT) models in the same scenario settings.

It is noted that in the text files containing the Lincoln model scenarios, although the initial conditions and the trajectories are provided, it is not clear what the destinations of the aircraft are. Therefore, while regenerating an individual Lincoln scenario using the proposed game theoretical method, we created three different trajectory types based on different assumptions about the destinations. These different types are explained below for a given encounter in the Lincoln Lab data set:

1. After Pass: The endpoint, the points where the trajectory end in the Lincoln encounter, is projected 5s into the future by using the final velocities given in the dataset, and this projected point is assumed to be the destination of the aircraft.
2. **Initial Direction**: The destinations of the aircraft is taken as the point which is reached in 100s with initial heading and velocity of the aircraft.

3. **Goal Direction**: Destination point is assumed to be the point that can be reached by starting in the initial position with the initial velocity and flying for 100s towards the endpoint of the Lincoln encounter.

It is noted that the above destination assignments are used only to determine the goals (italic) or endpoints (italic) of the proposed models. Once the goals are determined, the trajectories are formed based on the underlying game-theoretical decision-making algorithm. Three types of trajectories created using these three different destination assignments are presented visually in the following sections.

### 4.3.1. Fréchet Distance Results

Fréchet distances between the aircraft trajectories modeled by the game-theoretical (GT) and the Lincoln model are presented in Table 1 for the closest 25%, 50%, 75%, and 100% of all encounter scenarios provided in `cor_acl.txt` and `cor_ac2.txt`.

![Table 1](image)

Table 1: Fréchet distances between the trajectories created with GT and Lincoln models.

<table>
<thead>
<tr>
<th></th>
<th>Closest 25%</th>
<th>Closest 50%</th>
<th>Closest 75%</th>
<th>All Encounters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fréchet Distance (nm)</td>
<td>0.0815</td>
<td>0.1037</td>
<td>0.1421</td>
<td>0.2131</td>
</tr>
</tbody>
</table>

![Fig. 4](image)

Fig. 4. Results for 34 th encounter: (a,b,c) shows the Lincoln model trajectories, (d,e,f) shows the GT (1vs0) policy trajectories and (g) shows the separation distances of GT and Lincoln models. Trajectory labels are provided in subfigure g.
distances observed in the table show that the behavior of the proposed manned aircraft policies is similar to that of the Lincoln model. Among the 50 scenario comparisons, one of the “closest” results is presented in Fig. 4. The encounter scenario given in Fig. 4 is best modeled by the Goal Direction destination setting created by the GT method, using a Level-1 vs Level-0 encounter. A “medium” comparison result is presented in Fig. 5. Although this is considered as medium results, Fig. 5, shows that the GT generated trajectories are similar to the trajectories of the Lincoln model. The scenario presented in Fig. 5 is best modeled by the Goal Direction destination setting with a Level-1 vs Level-1 encounter. A comparison scenario that produced one of the “farthest” Fréchet distances between the trajectories created by the GT and Lincoln models is presented in Fig. 6. In this scenario, the GT policies try to keep a larger separation

Fig. 5. Results for 24th encounter: (a,b,c) shows the Lincoln model trajectories, (d,e,f) shows the GT (1vs1) policy trajectories and (g) shows the separation distances of GT and Lincoln models. Trajectory labels are provided in subfigure g.
distance than the Lincoln model. The scenario in Fig. 6 is best modeled by Level-2 vs Level-0 encounter with After Pass destination setting.

4.3.2. Euclidean Distance Results

Since Euclidean distance measure only allows comparisons between the trajectories with the same duration, the trajectories are

Table 2
Euclidean distances between the trajectories created with GT and Lincoln models.

<table>
<thead>
<tr>
<th>Euclidean Distance (nm)</th>
<th>Closest 25%</th>
<th>Closest 50%</th>
<th>Closest 75%</th>
<th>All Encounters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2663</td>
<td>0.3779</td>
<td>0.5326</td>
<td>0.8102</td>
</tr>
</tbody>
</table>
cropped to the duration of the shortest one. Aggregated Euclidean distance comparison results are presented in Table 2. As this table shows, for half of the encounters, the mean distance between trajectories generated by the GT and Lincoln models is 0.5 nm, which can be considered as a small distance in a 3D airspace.

A result that represents one of the “closest” match between the GT and Lincoln models is presented in Fig. 7. The scenario presented in Fig. 7 is modeled by a Level-1 vs Level-0 encounter with the destination setting of Initial Direction. A comparison result that indicate a
“medium” range match is presented in Fig. 8. The scenario presented in Fig. 8 is best modeled by Level-1 vs Level-0 encounters with the destination setting of After Pass. A scenario where the GT and Lincoln models differ more than the case given in Fig. 8 is presented in Fig. 9. Encounter scenario given in Fig. 9 is best modeled by Level-1 vs Level-0 encounters with After Pass destination setting.

When all the comparison results are considered, using both the Fréchet and the Euclidean metrics, it is seen that the proposed GT models provide reasonably similar results with the data-validated Lincoln model.
5. Simulation Results

As a demonstration of the proposed 3D airspace model capabilities, in this section, a quantitative analysis of multiple UAS integration in a crowded airspace is presented. Before presenting these results, single encounter scenarios, where two manned aircraft are on a collision path, are investigated to shed some light on the decision-making process of the pilots.

5.1. Single Encounter Scenarios of Manned Aircraft

The first scenario consists of two manned aircraft flying towards each other at different altitude levels. Aircraft are in a collision path, which means that if neither of the aircraft deviate from their trajectory, a conflict occurs. The initial horizontal and vertical separation distances between aircraft are 21 nm and 1000ft, respectively. The time horizon of the pilots is 20s, which means that pilots can oversee a conflict in a 20s time window prior to a separation violation. The distance horizon is 5 nm. Four simulations are conducted where, Level-1 vs Level-0, Level-1 vs Level-1, Dynamic Level-1 vs Dynamic Level-1 and Level-2 vs Level-1 pilots are on a collision path. The simulation results are given in Fig. 10, where (a)-(d) depicts the 2D projection of the encounters on the horizontal plane for different level settings, and (e)-(f) shows the corresponding separation distances. Manned aircraft are represented with squares that are color-coded for the pilot levels: Black, red and green squares represent level-0, level-1 and level-2 pilots, respectively.

Fig. 9. Results for 37th encounter: (a,b,c) shows the Lincoln model trajectories, (d,e,f) shows the GT (1vs0) policy trajectories and (g) shows the separation distances of GT and Lincoln models. Trajectory labels are provided in subfigure g.

5. Simulation Results

As a demonstration of the proposed 3D airspace model capabilities, in this section, a quantitative analysis of multiple UAS integration in a crowded airspace is presented. Before presenting these results, single encounter scenarios, where two manned aircraft are on a collision path, are investigated to shed some light on the decision-making process of the pilots.

5.1. Single Encounter Scenarios of Manned Aircraft

The first scenario consists of two manned aircraft flying towards each other at different altitude levels. Aircraft are in a collision path, which means that if neither of the aircraft deviate from their trajectory, a conflict occurs. The initial horizontal and vertical separation distances between aircraft are 21 nm and 1000ft, respectively. The time horizon of the pilots is 20s, which means that pilots can oversee a conflict in a 20s time window prior to a separation violation. The distance horizon is 5 nm. Four simulations are conducted where, Level-1 vs Level-0, Level-1 vs Level-1, Dynamic Level-1 vs Dynamic Level-1 and Level-2 vs Level-1 pilots are on a collision path. The simulation results are given in Fig. 10, where (a)-(d) depicts the 2D projection of the encounters on the horizontal plane for different level settings, and (e)-(f) shows the corresponding separation distances. Manned aircraft are represented with squares that are color-coded for the pilot levels: Black, red and green squares represent level-0, level-1 and level-2 pilots, respectively.
Fig. 10. First single encounter scenario trajectories and separation distances for Level-1 vs Level-0, Level-1 vs Level-1, Dynamic Level-1 vs Dynamic Level-1 and Level-2 vs Level-1.

Fig. 11. Second single encounter scenario trajectories and separation distances for Level-1 vs Level-0, Level-1 vs Level-1, Dynamic Level-1 vs Dynamic Level-1 and Level-2 vs Level-1.
Black solid line segments that are attached to the squares show the direction of the aircraft. Initial positions and the goals of the aircraft are represented with circles and triangles, respectively. It is noted that the “goals” on the figures are the points on the path from the initial positions to the destinations that are too far away to show in the snapshot. Dashed lines show the predicted trajectories of the aircraft. The two neighboring grid cells are 5 nm away. As Fig. 10 shows, no separation violations occur during flights (as stated earlier, a separation violation occurs when both horizontal (5 nm) and vertical (1000 ft) separation thresholds are broken). It is also observed that the dynamic level-1 policies do not change their levels since no conflict is overseen, as Figs. 10 (b) and (c) reveal.

The second scenario also consists of two manned aircraft in a conflict path flying towards each other. In this scenario, initial separations are 18 nm and 1000 ft, in the horizontal and vertical directions, respectively. Similar to the first scenario, the time horizon of the pilots is 20 s, and the distance horizon is 5 nm. We again simulate the scenarios where Level-1 vs Level-0, Level-1 vs Level-1, Dynamic Level-1 vs Dynamic Level-1 and Level-2 vs Level-1 pilots are on a collision path. Simulation results are presented in Fig. 11. Sub-Figs. 11 (a)-(d) show the 2D horizontal projection of the encounters and (e)-(f) depict the separation distances on the horizontal and the vertical planes. As seen from the figure, there is no instance of separation violation. However, during the dynamic level-1 encounter, one of the pilots increases his/her level in order to prevent a conflict. The yellow square on (c) shows the position when a conflict is detected, and the green square depicts the point where it is understood that the conflict persists and the level is changed to the best response to the level of the other pilot.

Similarly, the third scenario consists of two manned aircraft in a conflict path flying towards each other. In this scenario, initial separations are 8.72 nm and 1000 ft, in the horizontal and vertical directions, respectively. The time horizon of the pilots is 20 s, and the distance horizon is 5 nm. The scenarios where Level-1 vs Level-0, Level-1 vs Level-1, Dynamic Level-1 vs Dynamic Level-1, and Level-2 vs Level-1 pilots are on a collision path are simulated. Simulation results are presented in Fig. 12. SubFigs. 12 (a)-(d) show the 2D horizontal projection of the encounters and (e)-(f) depict the separation distances on the horizontal and the vertical planes. Different from previous two single-encounter cases, in the scenario where Level-1 vs Level-1 pilots are on a collision path, a separation violation is observed. In order to prevent this, in the dynamic level-1 encounter, one of the pilots changes his/her level to level-2. In subFig. 12 (c), the yellow square presents the position when a conflict is detected, and the green square shows the point where the level is changed since it is understood that the conflict is persistent.

### 5.2. UAS in a Crowded Airspace

In this section, an example use case for the proposed modeling framework is demonstrated in a 196,850.39 ft (length) x 98,425.97 ft airspace.
Fig. 13. Scenario with 200 manned aircraft and three UASs (cyan). Level-0, level-1 and level-2 pilots are represented by black, red and green squares, respectively. Yellow dots are UASs waypoints and black lines represent the ideal trajectory of the UAS in the center.

(a) Number of UAS separation violations. (b) Average UAS trajectory deviations.

(c) Average UAS flight times. (d) Average manned aircraft trajectory deviations.

Fig. 14. Responsibility analysis results for 200 manned aircraft and 3 UAS.
The outcomes of different separation responsibility assignments between the UAS and manned aircraft. Separation responsibility assignment is an important issue in addressing the integration of UAS into NAS (Johnson et al., 2012): it is crucial to determine which of the agents (manned aircraft or UAS) will take the responsibility of the conflict resolution.

In the scenario, initial positions, altitudes, and headings of aircraft are randomly assigned with the constraint that the horizontal and vertical distances between aircraft should be larger than 5 nm and 1000 ft, respectively. Velocities of all aircraft are set to 340 knots. The manned aircraft pilots make decisions based on the proposed pilot decision models.

Three UASs are placed in this airspace, with pre-programmed circular flight plans consisting of 4 waypoints. A snapshot of this scenario, projected on the 2D horizontal plane, is presented in Fig. 13, where black, red and green aircraft have level-0, level-1 and level-2 pilots, respectively. The aircraft that are colored with cyan are UASs, and yellow dots are UASs’ waypoints (only one set of waypoints is shown for clarity). The black lines that connect the waypoints form the ideal trajectories of UASs, which are to be followed unless a conflict is detected by the UAS SAA algorithms (see Section 2.2 for the description of these algorithms). If a conflict is detected, UAS perform evasive maneuvers dictated by their SAA algorithms and then turn back to their trajectory once the conflict is resolved. At the start of each simulation, level-0, level-1, and level-2 pilots are randomly assigned, with the constraint that level-0 DMs, level-1 DMs, and level-2 DMs constitute 10%, 60%, and 30% of the overall population. However, during the simulation, these numbers can change due to the dynamic level-k algorithm. The distribution of the levels is inspired from human experimental studies discussed in Costa-Gomes et al. (2009), where it is stated that in general level-0 frequency is small with most weight on level-1 and level-2. However, this distribution may not necessarily reflect the true distribution of levels among pilots. If needed, the proposed modeling framework allows different distributions.

Remark 3. The scenario described above is created to produce a diverse set of encounters. The proposed modeling framework has enough degrees of freedom to create different types of scenarios that accommodate further automation and more structured flights with strictly defined routes and procedures.

Three different responsibility assignment cases are investigated: 1. Manned aircraft are responsible, 2. Both manned aircraft and UAS are responsible and 3. UAS is responsible. In the first case, where only the manned aircraft are responsible, UASs follow their pre-planned path without using any SAA system, and manned aircraft act as dynamic level-0, level-1, and level-2 decision-makers. In the case where both manned aircraft and UASs are responsible, UAS executes the maneuvers dictated by their SAA algorithms, and manned aircraft follows dynamic level-k policies. In the last case, when only the UASs are responsible, manned aircraft follow their path without any evasive maneuvers, and UASs utilizes their SAA systems. The results of the scenarios are reported for two different SAA systems, SAA1 and SAA2 (see Section 2.2 for the SAA algorithms).

The outcomes of the aforementioned scenario are presented in Fig. 14, where (a) presents the total number of separation violations between manned aircraft and UAS, (b) shows the average trajectory deviation of UAS, (c) shows the average flight time of UAS and (d) presents the average trajectory deviation of manned aircraft. These averages (and error bars) are calculated over 500 simulations. Fig. 14-(a) shows that for both SAA systems, SAA1 and SAA2, the total number of separation violations is minimized when both UAS and manned aircraft are responsible for conflict resolution. Furthermore, it may be stated that in terms of safety, SAA1 performs better than SAA2 due to a lower number of separation violations. However, as Fig. 14-(b) indicates, SAA1 deviates from the ideal trajectory more than SAA2. Fig. 14-(c) demonstrates that SAA1 flight times are more than that of SAA2. This is expected since SAA1 trajectory deviations are larger. Finally, as seen from Fig. 14-(d), for both of the SAA algorithms, the manned aircraft trajectory deviations are smaller when the responsibility is shared, compared to the case where only the manned aircraft are responsible. These results signify that a distributed responsibility case is safer compared to either only UAS is responsible or only manned aircraft are responsible cases. It is noted that with the help of the introduced 3D airspace model, other partial responsibility cases can also be simulated.

5.3. Sensitivity Analysis

To analyze the effects of the weight parameters on the behavior of the overall system, different pilot models are trained by changing the “weight ratio”. The weight ratio, \( w_{\text{r}} \), is defined as the ratio of the performance weight terms, \( w_1 \) and \( w_3 \), to the safety weight terms, \( w_1 \), \( w_2 \) and \( w_3 \). Thus, \( R_{\text{w}} = \frac{w_1 + w_3}{w_1 + w_2 + w_3} \). The nominal value of the weight ratio, which is used for training the pilot models that are used to create the simulation results so far, is defined as \( k \). In order to understand the effect of the change in \( R_{\text{w}} \), new pilot models, i.e. level-1, level-2 and level-3 policies, are trained with \( R_{\text{w}} = k/2 \) and \( R_{\text{w}} = 2k \). The scenario presented in Section 5.2 is resimulated with these new policies. For the case in which both manned aircraft pilots and UAS are responsible for conflict resolution, UAS separation violations are investigated. The results are presented in Fig. 15-a: The number of UAS separation violations increases when the weight ratio increases for both sense and avoidance algorithms. Furthermore, the same analysis is conducted for different traffic densities and the results are presented in Figs. 15-b and -c. It is seen that the difference between the pilot models tends to effect the separation violation numbers more as the traffic gets more congested. This is observed for both of the SAA algorithms.
6. Conclusion

In this paper, a combination of the level-k reasoning game-theoretical concept and an approximate reinforcement learning method called Neural Fitted Q-learning is used to create a three-dimensional (3D) airspace modeling framework for predicting the possible outcomes of integrating Unmanned Aircraft System (UAS) into the National Airspace System (NAS). Compared to the earlier results of the authors, the assumption that the decision-makers’ levels remain the same during interactions and the requirement of keeping a large Q-value table are removed. These are achieved by the introduction of a dynamic level-k reasoning method and the employment of the Neural Fitted Q-learning algorithms, respectively. These improvements made it possible to model a larger class of interactions between the decision-makers, and this is demonstrated by simulating various single encounter scenarios in a 3D airspace. The proposed modeling framework can be used to quantitatively investigate how the safety and performance of the simulated airspace system are affected by the various integration technologies and concepts such as airspace density, minimum separation distance, and various UAS sense and avoid algorithms and their design parameters. One of the issues about UAS integration is the responsibility assignment during conflicts and it is shown how the 3D game-theoretical modeling framework discussed in this paper can be used to study this problem.

CRediT authorship contribution statement

Berat Mert Albaba: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft, Validation, Writing – review & editing. Negin Musavi: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft. Yildiray Yildiz: Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.
Acknowledgements

This effort was sponsored by the Scientific and Technological Research Council of Turkey under Grant No. 114E282.

References


