Annual Reviews in Control 48 (2019) 1–21

Contents lists available at ScienceDirect

Annual Reviews in Control

journal homepage: www.elsevier.com/locate/arcontrol

Vision article

Modeling cyber-physical human systems via an interplay between reinforcement learning and game theory

Berat Mert Albaba, Yildiray Yildiz*

Department of Mechanical Engineering, Bilkent University, Turkey

ARTICLE INFO

Article history:
Received 31 July 2019
Revised 8 October 2019
Accepted 9 October 2019
Available online 30 October 2019

Keywords:
Cyber-physical human systems
Game theory
Reinforcement learning
Model validation

ABSTRACT

Predicting the outcomes of cyber-physical systems with multiple human interactions is a challenging problem. This article reviews a game theoretical approach to address this issue, where reinforcement learning is employed to predict the time-extended interaction dynamics. We explain that the most attractive feature of the method is proposing a computationally feasible approach to simultaneously model multiple humans as decision makers, instead of determining the decision dynamics of the intelligent agent of interest and forcing the others to obey certain kinematic and dynamic constraints imposed by the environment. We present two recent exploitations of the method to model (1) unmanned aircraft integration into the National Airspace System and (2) highway traffic. We conclude the article by providing ongoing and future work about employing, improving and validating the method. We also provide related open problems and research opportunities.

Contents

1. Introduction . . . 2
2. Review of existing work . . . 3
   2.1. Unmanned aircraft systems integration into the national airspace . . . 3
   2.2. Road transportation . . . 4
3. The basics . . . 4
   3.1. Game theory . . . 4
   3.2. Reinforcement learning . . . 5
      3.2.1. Q-learning . . . 6
      3.2.2. Neural fitted Q-learning . . . 6
      3.2.3. Jaakkola reinforcement learning . . . 6
4. Bringing the pieces together: an interplay between game theory and reinforcement learning . . . 7
5. Hybrid airspace modeling . . . 8
   5.1. Observation and action spaces . . . 8
   5.2. Reward function . . . 9
   5.3. Physical models of manned and unmanned aircraft . . . 10
   5.4. Simulation results . . . 10
   5.5. Validation . . . 11
6. Road traffic modeling . . . 12
   6.1. Driver observation and action spaces . . . 12
   6.2. Reward function . . . 12
   6.3. Physical models . . . 13
   6.4. Training, average rewards and entropy . . . 13
   6.5. Data validation . . . 14

* Corresponding author.
E-mail address: yyildiz@bilkent.edu.tr (Y. Yildiz).

https://doi.org/10.1016/j.arcontrol.2019.10.002
1367-5788/

6.5.1. Validation with $n_{limit} = 1$ . . . 16
6.5.2. Validation with $n_{limit} = 3$ . . . 17
6.5.3. Validation with $n_{limit} = 5$ . . . 18
7. Computational complexity . . . 18
8. Ongoing and future work . . . 19
8.1. 3D hybrid airspace . . . 19
8.2. Large scale cyber-security scenarios . . . 19
8.3. Data validation . . . 19
9. Open problems and research opportunities . . . 19
10. Summary . . . 20
Declaration of Competing Interest . . . 20
Acknowledgements . . . 20
References . . . 20

1. Introduction

In a 2006 NASA report, allocation of tasks, as well as switching, between humans and automation is stated as one of the *highest priority research needs* for a successful next generation airspace development, where several new automation components are expected to be introduced to cope with the inevitable increase in traffic density (Sheridan, Corker, & Nadler, 2006). A 2012 U.S. Department of Defense report (Murphy & Shields, 2012) declares that the taxonomy established by the "levels of autonomy" creates "a focus on machines, rather than on the human-machine system," which in turn "has led to designs that provide specific functions rather than overall resilient capability". The same report suggests that *human-system collaboration* should be the defining theme for the design and operation of autonomous systems. In a 2017 review article (Lamnabhi-Lagarrigue et al., 2017), where an in-depth analysis is provided about the current and future roles of the systems and control field, the question "how to optimally conjugate automated systems with the interplay of humans?" is posed as one of the grand challenges. It is therefore clearly understood by engineers and scientists that considering humans as integral parts of complex physical systems that embody communication, computation, control and networking technologies, can potentially accelerate the advancement of technology that can address pressing human needs. This requires a new perspective that considers the human, the physical plant and enabling cyber-technologies as a single system, namely a cyber-physical and human system (CPHS). This is in contrast to imagining the human solely as a user who is isolated from the technology.

Adopting the CPHS framework brings its own challenges, especially when it comes to obtaining predictive models. Apart from the intricacies of cyber-physical system (CPS) modeling, the human element is usually the most demanding component of a CPHS in terms of forecasting the future behavior. The difficulty intensifies when the system contains more than one human, which requires factoring in multiple human-human and human-automation interactions. The focus of this paper is on the latter type, where human interactions are an inseparable feature of the system. To simplify the exposition in this paper, we use "human–machine interactions", "human-autonomy interactions" and "human interactions" interchangeably, although a more careful use of the language that pays attention to nuances is possible. To obtain realistic models of CPHS, therefore, we need to concentrate on modeling methods that give us the ability to include typical human characteristics. For example, human interactions are generally not deterministic in nature, which can be captured in CPHS models by utilizing a probabilistic modeling framework. Furthermore, before taking an action, a human generally contemplates other intelligent agents' (such as other humans or automation) possible actions and then tries to choose a move that will increase the chances of obtaining the best outcome reflecting his or her preferences. A representational CPHS model needs to incorporate this "strategic behavior" (Camerer, 2011) of humans. Finally, as much as we want to believe otherwise, humans do not always, if at all, act in an optimal fashion. Our cognitive capabilities and computational powers are not always ample enough to provide the best response to a given situation. This final point is important for distinguishing CPHS models from autonomy models, where, based on available information, an algorithm can possibly be designed to react in the most appropriate way. Models that comprise these three attributes of human reactions, namely, being probabilistic, strategic and non-optimal, have a higher chance of success in terms of representing real-life behavior, for the cases where multiple humans are involved.

A solution for CPHS modeling that addresses human interactions is proposed by the introduction of the "Semi Network-Form Games" (SNFG) formalism (Lee & Wolpert, 2011; Lee et al., 2013). SNFG merges three modeling tools: Bayesian networks, game theory (GT) and reinforcement learning (RL). While the Bayesian networks form the probabilistic foundation of the method, game theory provides the required mechanism to produce strategic behavior expected from human interactions, and reinforcement learning enables obtaining time-extended scenarios where humans take successive actions. Non-optimal behavior, which is thought to be a typical human trait, emerges naturally in SNFG with the type of the exploited game theoretical solution technique and reinforcement learning.

The first research result that exploits the SNFG idea of merging GT and RL to create a CPHS modeling framework for a realistic engineering system, including more than 2 humans, has appeared in Musavi, Onural, Gunes, and Yildiz (2016), where the problem of integrating unmanned aircraft systems (UAS) into the national airspace (NAS) is carefully studied. The results reported in Musavi et al. (2016) are supported by extensive simulation capabilities, where the interactions of 180 manned aircraft pilots are modeled. This study builds upon earlier initial attempts to model smaller airspace scenarios with human interactions (Yildiz, Agogino, & Brat, 2013; 2014; Yildiz, Lee, & Brat, 2012). Simultaneous modeling of a large number of decision making agents (pilots, in this case) in a complex scenario is difficult due to the computational cost. A common approach in the literature is to model a single decision maker, whose actions are of interest, and force the rest to obey certain kinematic and dynamic constraints, to obtain a reasonable model behavior. However, although this approach presents some insight into the dynamics of the system, it is limiting due to grossly simplifying the real-life human interactions. The study conducted by Musavi et al. (2016) provided, for the first time in the literature, probabilistic outcomes of UAS integration scenarios, where each of the 180 aircraft pilots are modeled using the game theoretical decision making process, simultaneously. In this research, Bayesian Networks are not utilized and the

probabilistic decision making is obtained through employing a stochastic reinforcement learning algorithm proposed in Jaakkola, Satinder, and Jordan. (1994). Recently, this work is extended for the cases where the aircraft can move both horizontally and vertically in a 3-dimensional airspace (Musavi, Manzoor, & Yildiz, 2018). This extension required a dramatically larger observation space for the pilots which ruled out the possibility of using exact RL methods. To address this issue, Neural-Fitted Q-iteration (Gabel, Lutz, & Riedmiller, 2011; Riedmiller, 2005; Riedmiller, Montemerlo, & Dahlkamp, 2007) is integrated into the game theoretical framework, which uses neural networks (NN) for compactly estimating the exact state-action values needed by the RL algorithm.

Another study, exploiting the same approach, conducted by Li et al. (2018) achieved a similar result in the automotive domain, by creating a modeling framework for road traffic consisting of 50 manned vehicles and an autonomous car. This result was a continuation of leading studies conducted by Oyler, Yildiz, Girard, and Kolmanovsky (2016) and Li et al. (2016). Similar to Musavi et al. (2018, 2016), this contribution is also the first in the automotive literature where a large number of decision making drivers are simultaneously modeled using a game theoretical modeling approach. Recently, an extended version of this work, which covers a larger class of interaction scenarios, with the help of a road traffic simulation on a 5-lane highway, is presented in Albaba, Yildiz, Li, Kolmanovsky, and Girard (2019), where validation studies by processing real traffic data, which is provided in Colyar and Halkias (2007), are conducted.

In this paper, we first present the basic components of the modeling approach discussed above and then provide the examples of CPHS framework creation, using GT and RL, for engineering systems that can be employed in predicting the outcomes of having several humans, automation and physical systems interact with each other in extended periods of time. These ideas exist in the literature in a fragmented manner, and by elucidating them in an aggregated form here, we provide a concise single source. The deliberations in this article on employing game theory and reinforcement learning for building CPHS modeling frameworks should benefit control practitioners whose goal is to obtain models of engineering systems where humans are active players. Finally, we discuss ongoing and future work about the topic, together with open problems that may provide several different research opportunities for the CPHS community.

The organization of the paper is as follows: In Section 2, we review the existing work. In Section 3, we explain the basic building blocks of the game theoretical model. In Section 4, we show how these blocks are combined together to form the overall modeling approach. In Sections 5 and 6, we present the exploitation of the approach to create models of two different engineering systems containing multiple human interactions. We provide a computational complexity analysis in Section 7. In Section 8 and 9 we discuss ongoing and future work, and related open problems, respectively. Finally, we provide a summary of the article in Section 10.

2. Review of existing work

In this section, we review research activities looking for the answer to this question: “How can we predict the outcomes of engineering scenarios involving a cyber-physical human system with not one but several human elements?”. The literature revolving around this salient question, either partially or completely, covers a wide range of engineering realms, which is hard to exhaustively examine in the limits of this article. Since, up until now, there has been two main studies addressing this question by exploiting the game theoretical modeling approach elaborated in this paper, we will focus on areas that are in the scope of these two research efforts: Unmanned aircraft system integration into the national airspace and road transportation.

2.1. Unmanned aircraft systems integration into the national airspace

Although unmanned aircraft systems (UAS) has been attracting increasingly more attention, we have not yet witnessed the maturation of the civil markets. One of the main reasons for this underutilized potential can be attributed to the lack of regular access to the National Airspace System (NAS) (Dalamagkidis, Valavanis, & Piegl, 2008). Due to the well-justified risk-averse nature of the aviation industry, advances in developing rules and procedures for UAS integration into NAS are progressing relatively slowly, which results in UAS flying mainly in restricted airspace. Until it is clearly assured that UAS will not pose a danger to the existing air traffic and thus their integration is proven to be safe, routine access to NAS will not be realized (European RPAS Steering Group, 2013; FAA, 2013). There are many studies that address the problem of UAS integration into the airspace in terms of providing methods and tools to ensure safety. Ding, Tomlin, Hook, and Fuller (2016) proposes an autonomous decision making system for UAVs for determining a safe landing site in the case of an anomaly. For UAV platoons, controller design frameworks are presented in Chen, Hu, Mackin, Fisac, and Tomlin (2015), Chen, Shih, and Tomlin (2016), Chen et al. (2017b), where collisions are eliminated and target states are reached. In Chen, Bansal, Tanabe, and Tomlin (2017a), collision-free trajectories are designed for large-scale multi-UAV systems in the presence of disturbances in vehicle dynamics, and the method is demonstrated using up to 200 UAVs, in simulation environment.

UAS integration still remains to be a challenge (Melnyk, 2019) and since we don’t have the necessary experience to evaluate the effects of integration and there is not enough data yet, the only way to predict the outcomes of adding UAS into the airspace is conducting careful simulations (DeGarmo, 2004). To obtain reliable simulation results, a high-fidelity model of the airspace in the presence of manned and unmanned aircraft together with their interactions need to be obtained.

A typical approach in the airspace traffic modeling literature includes the assumption of pilots always following an ideal behavior pattern without any deviations. This is an unrealistic assumption since, as discussed earlier, a representative model should allow probabilistic human behavior. For example, it is well-documented that pilots may ignore controller’s commands or do not obey traffic collision avoidance system (TCAS) resolution advisories during emergency situations (Pritchett, 2010). In addition, it is reported that only 13% of pilot reactions agreed with the pilot model, which predicts a deterministic behavior, used for establishing TCAS algorithms (Kuchar & Drumm, 2007; Lee & Wolpert, 2011).

One of the main issues that needs to be solved for a safe integration is the development of a dependable sense-and-avoid (SAA) technology for the unmanned aircraft systems. It is not possible to mature this technology without testing its performance through simulations with reliable pilot models (Maki, Parry, Noth, Molinario, & Miraflor, 2012). There exist several SAA methods introduced in the literature, with validations conducted via simulations and experiments. Kuchar et al. (2004) carefully analyzed the utilization of TCAS as the SAA logic, and performed simulation tests using the aircraft encounter model developed by Kochenderfer, Espindle, Kuchar, and Griffith (2008). During the tests, pilot reactions were assumed to be known beforehand, based on the respective motions of the conflicting aircraft. In another study, Perez-Batlle, Pastor, Royo, Prats, and Barrado (2012) suggested maneuvers to solve UAS separation conflicts and tested these suggestions with simulations that contain pilots following the recommended maneuvers without any error. The performance of the SAA algorithm

proposed in Florent, Schultz, and Wang (2010) was assessed by simulations and experiments. For these assessments, the manned aircraft in separation conflict with the UAS were assumed to continue their motion unaffected, while the UAS were implementing the proposed SAA technique. Another example where predefined pilot action models are employed for evaluating various SAA algorithms can be found in Billingsley (2006).

The modeling framework for UAS integration into NAS that is discussed in this article differentiates itself from the above mentioned environments by providing a platform where pilot actions are not pre-determined but obtained through a decision making process by satisfying a utility function, or a "happiness function", that reflects pilot preferences. Furthermore, with the proposed approach, several pilot-pilot and pilot-UAS interactions (180 of them) can be modeled in time-extended scenarios, in a probabilistic manner, with the help of the convergence of game theory and reinforcement learning. The details of this framework are provided in Section 5.

2.2. Road transportation

We have reliable physical models of road vehicles that have high predictive power. However, the modeling problem quickly becomes difficult to handle when the task is modeling the vehicle together with the driver operating it. Furthermore, if the desired outcome is a model for traffic containing several vehicles, both manned and unmanned, obtaining answers turns out to be a real challenge. The main obstacle in this matter in question is the lack of accurate human interaction models.

Valid human interaction models in road traffic may prove themselves useful for two main tasks: Creating accurate traffic simulators that can be used for initial testing and tuning of autonomous car control algorithms, and designing autonomous vehicle control systems based on human way of driving, which can improve the passengers' comfort by making them feel as if a human driver is in control (Carvalho, Lefevre, Schildbach, Kong, & Borrelli, 2015).

There are several successful driver modeling studies in the literature. Real traffic data is employed to obtain Hidden Markov Model based driver models in Lefevre et al. (2014) and Lefevre, Carvalho, and Borrelli (2015). A semiautonomous vehicle control architecture is proposed in Vasudevan et al. (2012) and Shia et al. (2014), where the driver model is obtained, through k-means clustering, and used to inform the controller that produces corrections for driver inputs. Logical, if-then-else commands form the driver decisions in a modeling framework created by Salvucci, Boer, and Liu. (2001). A multi-agent simulator is employed to model lane changing in Hidas (2002). Lane changing behavior is modeled also in Kumar, Perrollaz, Lefevre, and Laugier (2013), where Bayesian filters and support vector machines are utilized to predict driver intent. There is another body of work on modeling drivers as controllers in a closed loop control system, examples of which can be found in Hess and Modjtahedzadeh (1990), Sharp, Casanova, and Symonds (2000), Treiber, Hennecke, and Helbing (2000), Salvucci and Gray (2004) and Ungoren and Peng (2005). A more recent example of feedback controller type driver modeling can be seen in Wakitani et al. (2018).

The proposed modeling framework for road traffic in this article has the following distinctions compared to earlier work: (1) the driver interaction models are scalable and a traffic scenario consisting of several vehicles interacting with each other can be modeled using a game theoretical approach; (2) driver behavior is obtained through employing a decision making process, instead of assuming a preset driver action model that is a function of time or states; (3) driver models are simultaneously strategic, meaning that they consider other intelligent agents (drivers, unmanned vehicles) possible actions and produce a response accordingly, based on a utility function representing their priorities. This is in contrast to modeling a single driver as a decision maker while assigning pre-determined trajectory profiles for the others. All the listed advantages are realized through the game theoretical modeling approach discussed in this article. There are other game theoretical driving modeling approaches reported in the literature such as the ones proposed by Yoo and Langari (2012) and Yoo and Langari (2013), where driver interactions are successfully modeled. On the other hand, unlike the method discussed in this article, these studies do not consider a time-extended scenario. Dextreit and Kolmanovsky (2014) also consider a game theoretical approach to model the interactions between the driver and the ego vehicle's power train. By penalizing the vehicle-driver system features of fuel consumption, emissions, battery state and operating conditions, this approach is demonstrated, via experiments, to perform better than the baseline controller, in terms of these system features, while still providing good drivability. However, the utilized game theoretical approach, namely Stackelberg solution, quickly becomes computationally intractable as the number of intelligent agents in the game increases, unlike the hierarchical game theoretical approach used in the method proposed in this article. The details of the proposed method are provided in Section 6.

3. The basics

In this section, we present the fundamental building blocks of the game theoretical modeling framework discussed in this article. These blocks are game theory and reinforcement learning. Below, we explain these pieces with a relatively narrow scope, using a semi-formal language to allow easy access, and with enough details that will enable understanding of the basic ideas necessary to grasp the following sections. There are several sources in the literature that can be used for formal introductions to these topics, such as Fudenberg and Tirole (1991) and Camerer (2011) for game theory, and Wiering and van Otterlo (2012) and Sutton and Barto (2018) for reinforcement learning.

3.1. Game theory

Game theory studies the interactions between strategic agents. A strategic agent is one that considers other agents' possible actions and their effects on the game while making his or her own decisions. The theory makes predictions about the outcomes of these interactions using precise mathematics.

Players in a game theoretical setting refer to the entities who can effect the game by their *moves* (or actions, or decisions). *Strategy* of a player defines the procedure based on which a player chooses his or her actions. A *solution concept* is a well established set of rules that are used to predict how a game will unfold. A *Nash equilibrium* is a solution concept, defined similarly to the equilibrium in system dynamics: When players have no incentive to deviate from their selected actions, the game is said to be in Nash equilibrium. This means that in Nash equilibrium, players choose their best actions against each others' actions. A typical example where Nash equilibrium can be observed is a game called the *Prisoner's Dilemma*. In this game, there are two prisoners, Prisoner A and Prisoner B. They are put in separate rooms so that they can not communicate. Both of them are provided with the following information: If Prisoner A confesses the crime, he will be released provided that Prisoner B denies the crime, which will cause Prisoner B serve 10 years in prison. Similarly, if Prisoner B confesses, he will be released provided that Prisoner A denies the crime, which will put Prisoner A in prison for 10 years. If they both deny, each will serve 3 years. If they both confess, each will serve 5 years in prison. We can represent this game in matrix form as shown in

Prisoner B / Prisoner A	Confess	Deny
Confess	-5, -5	0, -10
Deny	-10, 0	-3, -3

Fig. 1. Prisoner's Dilemma.

Fig. 1, where players' *payoffs* amount to the negative of the years to be served in prison. It is seen that, although resulting in a low overall payoff, (Confess, Confess) choice is the only Nash equilibrium since no player wants to change their move once they are in this state. It is important to note that Nash equilibrium is not always unique and there may be more than one Nash equilibrium depending on the game.

There are other equilibrium concepts such as *quantal response equilibrium* where instead of giving the best response to other players' actions, the players choose a probability distribution over their action space where actions with higher expected payoffs have higher probability of being played.

Not all solution concepts predict an equilibrium. For example, *level-k thinking* is a non-equilibrium game theoretical model of strategic interactions, which assigns different levels of *reasoning* for players (Costa-Gomes, Crawford, & Iriberri, 2009; Stahl & Wilson, 1995). In this model, the lowest level of reasoning is level-0, which represents non-strategic thinking, simply meaning that the players who reason at this level have a strategy that does not take into account other players' possible actions. A level-1 player, on the other hand, takes the best action assuming that his or her opponents are level-0 players. Similarly, a level-k player responds best to his *belief* that the other players are reasoning at level-(k-1). Therefore, this model assumes an *iterated* best response (Crawford, 2008). Experimental results presented in Camerer (2011) corroborate the predictions of this solution concept with varying success.

An elementary example for the level-k reasoning model can be given considering two people, named Diana and Ritchie, walking towards each other, along a collision path, in a university corridor. If Ritchie decides to continue walking without considering Diana's possible actions, Ritchie can be considered as a level-0, non-strategic thinker. On the other hand, if Diana believes that Ritchie is a level-0 thinker and therefore decides that the best action is stepping right, then Diana can be modeled as a level-1 player. Although there exists experimental evidence for level-k predictions, this simple example presents a difficulty in this approach: The players' beliefs of others may be wrong. Another issue is that even if the players correctly estimate their opponents' levels, the behavior patterns coded by different levels are sensitive to the selection of the level-0 algorithm. That's why level-0 is often referred as the *anchoring level*. Although these "problems" are real, they may actually be considered as the strengths of this method when it comes to modeling humans. As discussed in the Introduction section, in CPHS with multiple humans, to obtain reasonable predictions, we need models that do not foresee optimal behavior all the time. Another perspective is that level-k thinking represents the interactions between intelligent agents that do not have a long interaction history, therefore it is actually expected that their initial assumptions about each others' strategies may not fully reflect reality. This also helps obtain non-optimal human reactions that are providing best responses to their beliefs about the outside world.

3.2. Reinforcement learning

Reinforcement learning (RL) can be defined as a mathematical representation of learning through reward and punishment. To clarify this definition, and to be able explain the RL algorithm used in the CPHS modeling framework discussed in this paper, we first need to identify main elements of RL and make certain definitions that hold true for almost all RL methods. In RL, there exists an *agent* capable of exerting *actions* that can change the *state* of the environment where the agent operates. The RL problem can be defined as finding the optimal set of action sequence for an agent to achieve a given goal defined as a function of the environment states, through interaction with the said environment. For example, the problem may be making a mobile robot (agent) to go from point A to point B (goal) in a 10 by 10 grid-world with obstacles (environment) by deciding whether to move left, right, forward or backward (actions), in every step of the way. In this scenario, the state can be defined as the grid location the robot is occupying.

RL uses the idea of a *reward function* to describe the preferences of the agent (or the designer of the agent) while its learning to achieve a predetermined goal. In the mobile robot example given above, the reward can simply be defined as zero if the robot is at the goal state of point B, and a fixed negative number, otherwise. A *policy* is defined as a probabilistic map from states to actions. The task of the RL algorithm is to find a policy that will make the agent maximize a cumulative discounted reward, during the time of its operation. One way to express the cumulative reward is given as

$$C = \sum_{t=0}^{\infty} \gamma^t r_t, \tag{1}$$

where γ is the discount factor and r is the reward obtained in every step t. There are various RL techniques proposed to discover action sequences that will attain the goal of maximizing (1). Almost all of these different methods are based on estimating *value functions*, which can be regarded as the value of being in a certain state, based on the policy being implemented. A value function can be given as

$$V^{\pi}(s) = E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s \right\} \tag{2}$$

where π represents the implemented policy and s represents state. A similar function, the estimation of which also characterizes the RL method, stands for the value of taking a certain action a, in a given state s. This function is defined as

$$Q^{\pi}(s,a) = E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a \right\}. \tag{3}$$

The aim of RL is to find the optimum policy π^* that will maximize (optimize) the value function. The optimal value function is written as V^{π^*} and all its state values are larger than or equal to that of all other value functions that are created by policies different than the optimal policy. This can be represented as $V^{\pi^*}(s) \geq V^{\pi}(s), \forall \pi, \forall s$. Similarly, we can write $Q^{\pi^*}(s,a) \geq Q^{\pi}(s,a), \forall \pi, \forall (s,a)$, for the optimal action value function. Once the optimal action value function is found, the policy

$$\pi^*(s) = \arg\max_a Q^{\pi^*}(s,a) \tag{4}$$

can be used to select the best action in each state. The answer to the question "how to find the optimal value function" determines the type of RL algorithm. The process of finding the optimal policy is sometimes called *training*. Fig. 2 depicts the general training process of RL. In the figure, the agent observes the states and produces an action based on the observed states. This action influences the environment and results in a new set of states. These

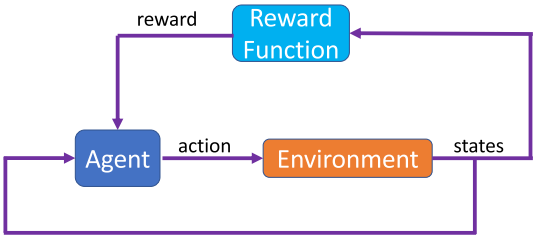


Fig. 2. Reinforcement learning process.

new states are evaluated by the reward function and a reward signal is formed. The agent uses this signal to update the policy that is being trained and the next cycle starts with the new action.

In the following subsections, we explain one of the most basic RL algorithms, *Q-learning*, and then present two other RL methods that are utilized in the game theoretic modeling framework elaborated in this article.

Remark 1. The RL algorithms used in this paper are used to obtain human response policies, which can then be used to quantitatively analyze scenarios where humans are involved, in the simulation environment. Therefore, there is no physical interaction with the environment during learning.

3.2.1. Q-learning

One of the RL methods that played a significant role for the success of RL is *Q-learning* (Watkins, 1989). In Q-learning, an incremental estimate of the optimal action value function is realized using the update rule

$$Q_{k+1}(s_t, a_t) = Q_k(s_t, a_t) + \alpha\left(r_t + \gamma \max_a Q_k(s_{t+1}, a) - Q_k(s_t, a_t)\right), \tag{5}$$

where α is the step size and γ is the forgetting factor. It is noted that Q-learning is indifferent to the policy that the agent uses during training to *explore* the environment, which means moving from one state to the other. This type of RL algorithms, where exploration and value function updates (or policy updates) are independent, are called *off policy* methods. It can be shown that the learned action value function Q converges to the optimal action-value function Q^* with probability 1, if all state-action pairs $(s,\ a)$ continue to be visited during training, and if the number of these visits converges to infinity. Convergence is achieved exactly if the step size parameter obeys a variant of the stochastic approximation conditions, given as $\Sigma_{k=1}^{\infty}\alpha_k = \infty,\ \Sigma_{k=1}^{\infty}\alpha_k^2 \leq \infty$. For a constant step size parameter, convergence is achieved in the mean if $0 \leq \alpha \leq 1$. It is noted that continuously visiting state-action pairs is a requirement for any algorithm that is developed for the general case, where a model for the environment is not provided, to achieve convergence to the optimal solution.(Sutton & Barto, 1998). The discount factor γ determines how valuable the future rewards are. For instance, when γ is 0, agent only tries to maximize the immediate rewards. As this value approaches to 1, the agent becomes more far-sighted.

3.2.2. Neural fitted Q-learning

Instead of keeping a table of Q values, some RL methods use compact structures that provide approximate Q values. This approach is especially useful when the state space is very large. Neural Networks (NN) are one of the effective tools used to store the Q-values in a compact manner, thanks to their universal approximation property. Unlike the conventional Q-learning method explained above, the state-action values are not stored in a table but instead, computed as the output of a function constructed by the specific NN structure: If a state-action pair is fed to the NN as the input, the corresponding approximate Q-value can be obtained as the NN output. The training of the NN can be achieved by first defining an error function reflecting the difference between the current and the target Q-values, and then minimizing this function by back-propagation. Although calculating the Q-values using a NN provides an effective method, it can fail, either completely or by requiring impractical convergence times, due to the global representation mechanism (Riedmiller, 2005): during the training process, NN weights are updated after the introduction of each individual state-action pair, which also effects the Q-values of other pairs. This may nullify the previous training gains in other regions. On the other hand, the global representation enables the generalization power of NNs by assigning similar Q-values to similar state-action pairs, and thus eliminating the need to train the NN for every possible pair. Therefore, a method is needed that can both exploit this property and eliminate its detrimental effects.

Algorithm 1 NFQ algorithm.

1: $k = 0$
2: Initialize the neural network
3: **while** $k < N$ **do**
4: Generate experience set $G = \{(input^i, target^i), i = 1, 2, \ldots, \#E\}$ where
5: $input^i = s^i, a^i$, where s^i is the state and a^i is the action of i^{th} experience,
6: $target^i = r^i + \gamma \min_a Q_k(s'^i, a)$, where r^i is the transition cost and $\gamma \min_a Q_k(s'^i, a)$ is the weighted expected maximum path reward for the next state s'^i.
7: Calculate the batch error as $\Sigma_{i=1}^{n}(Q^k(s_i, a_i) - target^i)^2$, where n refers to the experience set size.
8: Train the network to minimize the batch error, using resilient back-propagation and obtain Q_{k+1} .
9: $k+ = 1$
10: **end while**

Neural Fitted Q-learning, proposed by Riedmiller (2005), achieves to both employ the generalization power of NNs and prevent its potentially harmful effects by storing previous experiences in the form of 3-tuples, $(s,\ a,\ s')$, in which s is the original state, a is the action taken and s' is the reached state, and reusing these experiences whenever an update is performed after the introduction of a new data point. Calling a collection of these experiences as set E, the NFQ method is given in Algorithm 1. In the implementation of NFQ learning, it is advised that instead of a random collection of experiences, greedy search, using available Q-values, and random exploration are used together.

NFQ contains two types of hyper-parameters: parameter of the Q-learning, the forgetting factor γ, and the parameters used for NN training. The selection of the Q-learning hyper-parameter γ is explained in the previous section. For NN training, the resilient propagation algorithm proposed in Riedmiller and Braun (1993), which works well for batch learning, is suggested since NFQ is also based on batch learning. It is discussed in Riedmiller and Braun (1993) that different values of hyperparameters do not effect the performance of the algorithm dramatically and therefore some predefined values suggested in the paper can be used for most of the problems. In many NN training software packages (Abadi et al., 2015; Paszke et al., 2017) these values are already set as default so no further tuning is necessary.

3.2.3. Jaakkola reinforcement learning

An agent being trained by RL uses the available information from the environment. This information is generally called the

"*state*" of the environment. When the state of the environment contains all relevant information about the current and the past interaction dynamics between the agent and the environment, this state is said to have the Markov property (Sutton & Barto, 2018). A learning task involving interactions with an environment that has Markov property is called a Markov Decision Process (MDP). More specifically, defining the probability of transitioning from state "s" to state "s'" and obtaining a reward "r", given an action "a" as $P(s', r|s, a)$, if this probability depends only on "s" and "a" but not on earlier states and actions, this learning task is called an MDP. In almost all RL methods that have convergence guarantees, the underlying dynamics is assumed to be an MDP. In the aerospace and automotive application scenarios that are investigated in this paper, although the underlying dynamics are MDPs, the agents can realistically observe only a portion of the available states. Therefore, from the agents' point of view, the tasks are Partially Observable Markov Decision Processes (POMDP).

Jaakkola reinforcement learning algorithm (Jaakkola et al., 1994) is developed specifically for systems that can be modeled as POMDPs and therefore is a suitable RL method to be employed in the learning tasks that are discussed in this work. In Jaakkola algorithm, along with Q-function, the value function, V, is also used. At the beginning of the Jaakkola Algorithm, Q values are set to zero for each state-action pair. Moreover, for each state, probability distribution of actions is set to a uniform distribution. Then, for each iteration (s, a, s'), Q and V values are updated according to following equations:

$$\begin{aligned}
\beta_t(s,a) &= \left(1 - \frac{\chi_t(s,a)}{K_t(s,a)}\right)\gamma_t\beta_{t-1}(s,a) + \frac{\chi_t(s,a)}{K_t(s,a)} \\
\beta_t(s) &= \left(1 - \frac{\chi_t(s)}{K_t(a)}\right)\gamma_t\beta_{t-1}(s) + \frac{\chi_t(s)}{K_t(s)} \\
Q_t(s,a) &= \left(1 - \frac{\chi_t(s,a)}{K_t(s,a)}\right)Q_{t-1}(s,a) + \beta_t(s,a)(R_t - R) \\
V_t(s) &= \left(1 - \frac{\chi_t(s)}{K_t(s)}\right)V_{t-1}(s) + \beta_t(s)(R_t - R) \qquad (6)
\end{aligned}$$

where, s is the state, a is the action and t is the time step. Moreover, $\chi_t(s, a)$ ($\chi_t(s)$) is equal to 1 if the given state-action pair (state) is visited, and 0 otherwise; $K_t(m, a)$ ($K_t(s)$) is the number of times the state-action pair (state) is visited; R_t is the reward in time step t; R is average reward and γ_t is the discount factor. After the calculation of Q and V functions, Jaakkola algorithm updates its trained policy $\pi(a|s)$ using the update rule

$$\pi(a|s) = (1-\epsilon)\pi(a|s) + \epsilon\pi^1(a|s), \qquad (7)$$

where ϵ is the update rate, and $\pi^1(a|s)$, the policy that the trained policy is being changed towards, is a greedy-policy based on the calculated Q(s,a) values. In other words, $\pi^1(a|s)$=1 if the action "a" has the highest Q-value in a given state "s". It can be shown (Jaakkola et al., 1994) that this policy update always increases the average reward, unless the condition

$$\max_a[Q(s,a) - V(s)] > 0 \qquad (8)$$

is satisfied. The algorithm increases the average reward until the condition (8) is false, which constitutes a local maximum.

The Jaakkola algorithm consists of two hyper-parameters: the discount factor γ and the update rate ϵ. For convergence guarantees, γ should initially be selected as a number between zero and one, and should be scheduled in such a way that it converges to 1 in the limit. ϵ, on the other hand, should satisfy $0 \leq \epsilon \leq 1$.

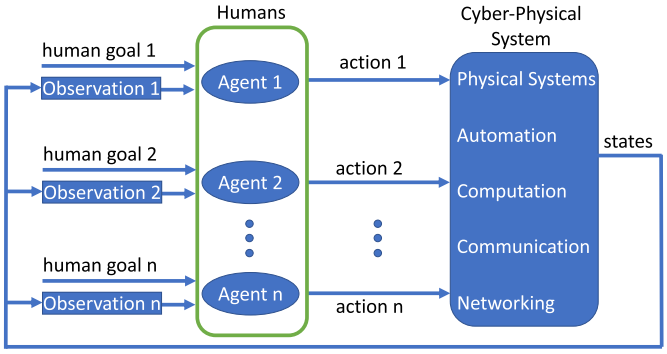


Fig. 3. Cyber-physical human system with multiple humans.

4. Bringing the pieces together: an interplay between game theory and reinforcement learning

A schematic of the cyber-physical human system (CPHS) we are interested in, involving multiple human interactions, is depicted in Fig. 3. This is a simplified diagram of the overall system, showing how humans' actions change the cyber-physical system states, which are observed by humans who produce the next set of actions and initiate the next cycle accordingly. It is noted that humans' actions are also affecting each other through the closed loop nature of the information flow. Another important take-away from this figure is that human observations are not necessarily the same and an individual human agent does not have full state information. *Observation i* blocks, where $i = 1, 2, \ldots, n$, represent the human observation process. This process is limited and imperfect, therefore each human receives a noisy subset of the whole system state. *Human goal* blocks represent what each of the human agent is trying to accomplish, which can be driving from point A to point B in a road traffic scenario, or protecting his or her own aircraft from a hacker attack in a cyber-security scenario.

Obtaining a model for the system shown in Fig. 3 requires considering the interaction of human actions. Humans are strategic thinkers and therefore if we want to build a model that represents reality, to the best of our ability, we need to find a way to have agent models that consider other agents' possible actions before making a move. As discussed in Section 3.1, this is what game theory is all about: Modeling the interactions between strategic agents. Therefore, we may utilize game theory to solve this problem. However, there is one complication in this approach. This is a system where the interaction may last for long periods of time, and therefore obtaining an equilibrium solution can be computationally intractable. If the number of agents get larger, as in several real world applications, even for short periods of interactions, the computational cost grows rapidly. One way around this problem is using the non-equilibrium game theoretical solution concept, level-k thinking, explained in Section 3.1, where human actions are predicted in an iterated manner, instead of being evaluated at the same time. This means that once the *anchoring level*, level-0, is selected, a level-1 human agent's behavior can be identified as the best response to all the other human actions that are determined by the level-0 policy. Similarly, once level-1 behavior is identified, all the agents in the system are assigned the level-1 policy except the one whose level-2 behavior is to be found. Therefore, to predict the policy of a level-k agent, all the rest of the agents' policies are set to level-(k-1), which effectively make them a part of the environment whose dynamics are known, and level-k policy is determined as the best response to the rest of the level-(k-1) policy actions (see Fig. 4). This isolates the level-k policy as the single policy that needs to be computed. It is noted that in Fig. 4, the

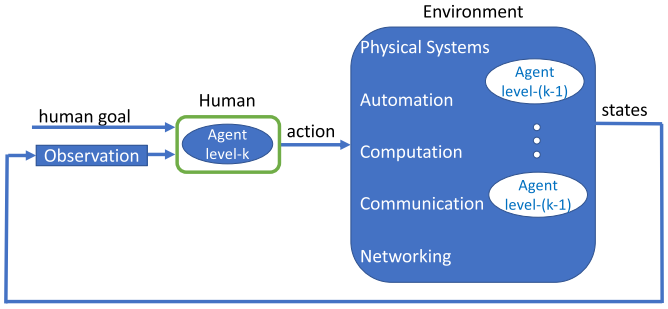


Fig. 4. Obtaining level-k policy.

actions of the level-(k-1) agents are not shown and the *states* still belong to the CPS. These details are omitted for brevity.

Once the difficulty of high computational cost due to multiple decision makers is solved using level-k thinking, the problem reduces to estimating the optimal action sequence of an intelligent agent in a given environment. This problem can be stated as a reinforcement learning (RL) problem by properly defining the states, actions, the environment and the reward function. (Compare Fig. 2 and Fig. 4.) Therefore, using one of the suitable RL methods described in Section 3.2, modeling of the CPHS with multiple human interactions can be completed. The type of the preferred RL algorithm depends heavily on the observation and action spaces of the agents, which in turn depend on the engineering domain of the CPHS operation. The general algorithm used in obtaining the agent policies, demonstrating the interplay between the game theory and RL, is provided in Algorithm 2, where k is the maximum desired level. It is noted that, a level-k agent can be made to best-respond to all lower levels, $k = 0, 1, \ldots (k-1)$, instead of only best-responding to level-(k-1), which may or may not be desired depending on the application, by including all lower levels in the agent's policy space.

In the following sections, based on Musavi et al. (2016) and Albaba et al. (2019), we explain how this interplay between RL and game theory is used to create modeling frameworks for two different engineering realms, where multiple (180 in one case and 125 in the other) human interactions are involved.

Remark 2. The methods discussed in this paper, which are used to model cyber-physical human systems, are used solely for modeling purposes. Therefore, they are not meant to be used in physical motion systems.

Algorithm 2 Interplay between RL and game theory.

```
Set i = 0
while i < k do
    Load the level-i policy
    Set cognition levels of all players in the environment other
    than the learning agent to level-i, i.e. set policies of players
    to level-i policy
    Place the learning agent in the initialized environment, in
    which all players are level-i
    Start the training of the learning agent using a reinforcement
    learning method, through which agent learns how to best
    respond to level-i players
    Once the training is completed, learning agent becomes a
    level-(i+1) player
    Save the policy of the learning agent as level-(i+1) policy
    i += 1
end while
```

Remark 3. There exist several successful cyber-physical system models in the literature. The bottleneck in driving models for cyber-physical human systems, is computing the multi-agent, multi-move decision making dynamics, which corresponds to obtaining the models in the green "Humans" block in Fig. 3. Therefore, in this paper, to emphasize the power of the discussed game theoretical modeling approach in predicting human responses, we avoided employing complicated vehicle models.

Remark 4. The cyber-physical human systems modeling method explained in this paper merges reinforcement learning and game theory. It is noted that the level-k approach is not the only game theoretical method that can be employed here. Theoretically, other game theoretical methods can also be used. However, as explained in this section, thanks to the hierarchical modeling approach inherent in the level-k reasoning, modeling multi-move, multi-agent scenarios with simultaneous decision makers can be handled in a computationally tractable manner, which makes the level-k method a suitable candidate for the scenarios investigated in this paper.

5. Hybrid airspace modeling

With *hybrid airspace* we refer to an airspace where manned and unmanned aircraft coexist. As discussed at length in Section 2.1, obtaining hybrid airspace models is a necessity for successful integration of unmanned aircraft systems (UAS) into the National Airspace System (NAS). In this section, we present how the game theoretic modeling framework discussed here is used to realize the modeling of these systems.

Fig. 5 shows a hybrid airspace scenario where a UAS (cyan) is assigned to follow certain waypoints (yellow) in a crowded airspace filled with manned aircraft (red). We are interested in how the overall system will evolve in time.

To be able to predict the possible outcomes of this scenario, we need to create a model that will capture the reaction dynamics of manned aircraft pilots in cases of separation conflicts. As discussed earlier, the game theoretical modeling approach produces policies, which are probabilistic maps from observations to actions, to represent human reactions. Therefore, to obtain these policies, we first need to clearly define the *observation* and *action spaces* and represent them in a way that is meaningful for the RL algorithm. Once these spaces are explicitly defined, pilot goals and preferences need to be expressed in form of a *reward function*. Furthermore, to train pilot policies in a simulation environment, we need to realize the motions of the aircraft, manned and unmanned, using their *physical models*. Finally, *sense and avoid algorithms* need to be integrated to UAS dynamics for collision avoidance. Below, we explain how these pieces are obtained and then assembled to create the overall hybrid airspace model. Furthermore, we discuss the validation studies conducted using data.

5.1. Observation and action spaces

Self-separation concept, where the pilots (and crew) are responsible for keeping a safe distance from encountered traffic, is a procedure that is being explored for Next Generation (NextGen) Air Transportation System (Wing et al., 2013). One of the technologies that can make this possible is Automatic Dependent Surveillance Broadcast (ADSB), which provides state information of the surrounding traffic, with a precision better than the radar (Kacem, Wijesekera, & Costa, 2018). In the above scenario, we assume that aircraft are equipped with this technology. To factor in cognitive limitations, the observation space is set as a pie shaped region formed by two circles sharing a common center, which is depicted in Fig. 6, on the left. The inner circle radius is taken as 1 nmi and

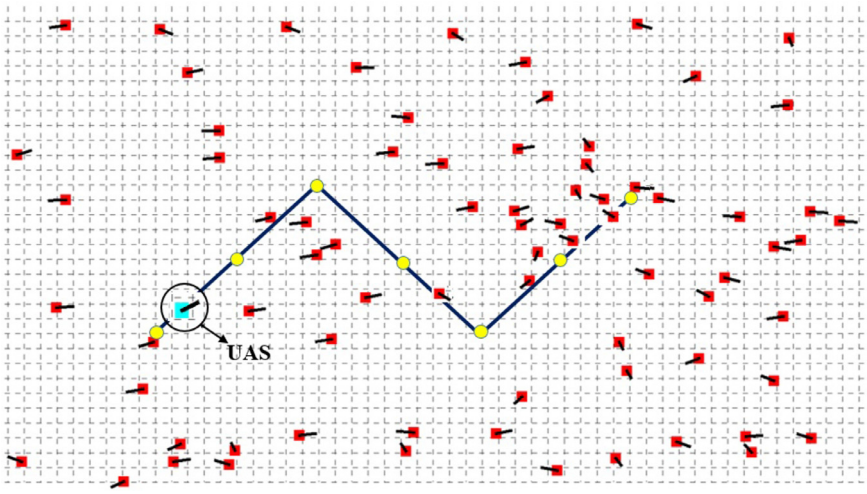


Fig. 5. A hybrid airspace scenario where red squares represent manned aircraft and the cyan square is used to indicate an unmanned aircraft (UA). The yellow circles in the 600 km × 300 km airspace are the waypoints assigned to the UA. (Musavi et al., 2016, reprinted with permission of the American Institute of Aeronautics and Astronautics, Inc.). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

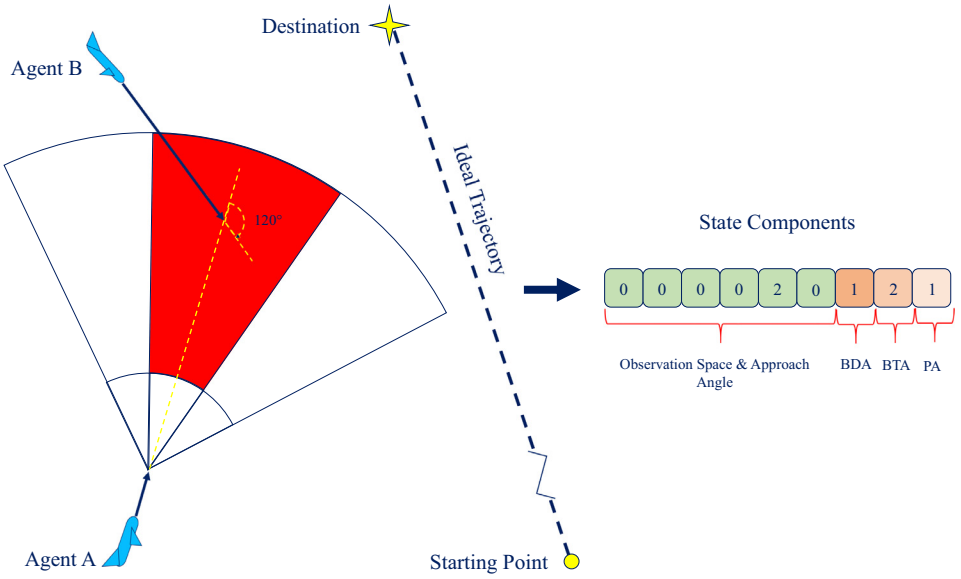


Fig. 6. Observation space. (Musavi et al., 2016, reprinted with permission of the American Institute of Aeronautics and Astronautics, Inc.).

the outer circle radius is taken as 5 nmi, reflecting the standard separation requirements for manned aircraft (Perez-Batlle et al., 2012). These circles are then divided into 3 slices resulting in 6 different observation regions. If an intruder approaches one of these regions, the region that is being approached is coded with number 1, 2, 3 or 4, depending on the approach angle, while the rest obtains a zero value. For example, in Fig. 6, Aircraft B is approaching Aircraft A's observation space with a 120° angle which makes the corresponding entry for the approached region take the value of 2, while the rest are assigned 0. This is shown under the *state components* section on the rightmost side of the figure. The information of the best immediate actions that will make the aircraft approach its predetermined trajectory (Best Trajectory Action, BTA) and to its predetermined destination (Best Destination Action, BDA), together with the pilot's Previous Action (PA) are also coded into the state components. The pilot action space consists of 3 actions: *45° left, straight* and *45° right*, and these actions are coded as 0, 1 and 2, respectively, in the state components. It is noted that these actions go through aircraft dynamics to produce the actual aircraft motion. During training, the state components entries are continuously updated, based on the explanations given above, and used in the RL algorithm. With the aforementioned observation and action spaces, the RL algorithm need to assign Q values to $5^6 \times 3^3 \times 3 = 1,265,625$ state-action pairs.

Remark 5. The actions described in this section are not the actions of the aircraft but the decision inputs to the real continuous aircraft dynamics, whose outputs are the aircraft actions. The aircraft dynamics is explained in Section 5.3.

5.2. Reward function

For the RL algorithm to evaluate the desirability of selecting a certain action given a state, a reward function that specifies pilot preferences and goals is required. For the UAS integration into NAS scenario, the reward function

$$r = -\omega_1 C - \omega_2 S - \omega_3 I + \omega_4 D + \omega_5 P - \omega_6 E \tag{9}$$

is used, where C and S represent the number of aircraft (manned or unmanned) occupying a space within the collision and separation regions of the ego aircraft, respectively. The definition of

these regions can be found at Planning et al. (2007) and Perez-Batlle et al. (2012). I is binary and gets the values of 1 or 0 depending on whether the ego aircraft is approaching or distancing from the intruder. D and P are the degrees of approach (or distancing) by the ego aircraft to its destination and to its defined trajectory, respectively, normalized by the distance covered in one time step. To accommodate the tendency of humans to minimize energy consumption as much as possible, E is introduced to represent *effort*, taking the values of 1 or 0, depending on whether the pilot takes a new action or not, respectively.

Remark 6. The selection of the reward function plays a crucial role in the performance of the reinforcement learning algorithm. A formal procedure for the determination of a proper reward function is an open area of research. However, addressing context-dependent problems that reflect the agent's preferences, while keeping the function simple is the general approach for determining the reward functions. One problem that needs to be solved in this section is avoiding collisions and separation violations, which is addressed by the terms C, S and I in (9). Another problem is reaching a given destination while following a predetermined trajectory, which is addressed by the terms D and P. Finally, the problem of energy conservation is addressed by the term E. An alternative to (9) could be a reward function that uses continuous variables for all the terms. This would increase the resolution but could unnecessarily complicate the function. In certain problems such as learning a policy from an expert, the reward function selection can be done using a more systematic approach by solving an inverse reinforcement learning problem (Sutton & Barto, 2018).

5.3. Physical models of manned and unmanned aircraft

As explained above, aircraft are controlled by pilots' commands of heading angle changes ($\pm 45^\circ$). Using the standard turn rate of 3 deg/s angular velocity (Nancy, 2016), we model the aircraft turning motion with a first order dynamics having a time constant of 10 s. The related differential equation is

$$\dot{\Psi} = -0.1(\Psi - \Psi_d), \tag{10}$$

where Ψ and Ψ_d represent the current and the desired heading angles, respectively. The manned aircraft are assumed to be flying in en route phase with a constant velocity v, having the x and y coordinate components

$$v_x = |v| \sin \Psi \tag{11}$$

and

$$v_y = |v| \cos \Psi. \tag{12}$$

Unmanned aircraft are assumed to fly autonomously while avoiding collisions using an onboard sense and avoid (SAA) algorithm. This algorithm commands velocity vector changes if an intruder is detected. Using a 1 second time constant (Mujumdar & Padhi, 2011) and a first order dynamics, the velocity vector dynamics are modeled as

$$\dot{\vec{v}} = -(\vec{v} - \vec{v}_d), \tag{13}$$

where $\vec{v}_d$ is the desired velocity vector. Two SAA algorithms are employed in the simulation environment with different velocity vectoring properties. Both of them first detects a probable conflict by forecasting the future trajectories of both the ego and intruder aircraft, and checking whether the minimum calculated distance, R, between them is smaller than a predetermined threshold value. If a conflict is detected, a desired velocity vector command, $\vec{v}_d$, is produced. One of the SAA algorithms (SAA1) proposed by Fasano et al. (2008) issues the velocity command

$$\vec{v}_d = \left(\frac{\vec{v}_{ei} \cos(\eta - \xi)}{\sin(\xi)} \left(\sin(\eta) \frac{\vec{v}_{ei}}{|\vec{v}_{ei}|} - \sin(\eta - \xi) \frac{\vec{r}}{|\vec{r}|} \right) + \vec{v}_i \right), \tag{14}$$

Algorithm 3 Interplay between RL and game theory in NAS.

1: $i = 0$
2: **while** $i < k$ (k is the maximum cognition level) **do**
3: Load the level-i policy
4: Set the policies of all the pilots in the scenario, other than the ego pilot (the pilot being trained), to level-i
5: Start the training of the ego pilot using reinforcement learning, through which the pilot learns how to best respond to level-i pilots
6: Once the training is completed, the ego pilot becomes a level-$(i+1)$ pilot
7: Save the policy of the ego pilot as level-$(i+1)$ policy
8: $i+=1$
9: **end while**

in cases of conflict, where the ego unmanned aircraft velocity command and intruder aircraft velocity are represented by $\vec{v}_d$ and $\vec{v}_i$, respectively. Similarly, $\vec{r}$ and $\vec{v}_{ei}$ refer, respectively, to the relative position and velocity between these two aircraft, and η is the angle between these two vectors. Angle ξ is determined as $\xi = \sin^{-1}(R/|\vec{r}|)$. The other SAA algorithm (SAA2) used in simulations provides the velocity command

$$\vec{v}_d = \frac{-\vec{v}_e \left(\frac{\vec{r}_0 \cdot \vec{v}_{ei}}{|\vec{v}_{ei}|} \right) - (R - |\vec{r}_m|) \frac{\vec{r}_m}{|\vec{r}_m|}}{\left| -\vec{v}_e \left(\frac{\vec{r}_0 \cdot \vec{v}_{ei}}{|\vec{v}_{ei}|} \right) - (R - |\vec{r}_m|) \frac{\vec{r}_m}{|\vec{r}_m|} \right|} \tag{15}$$

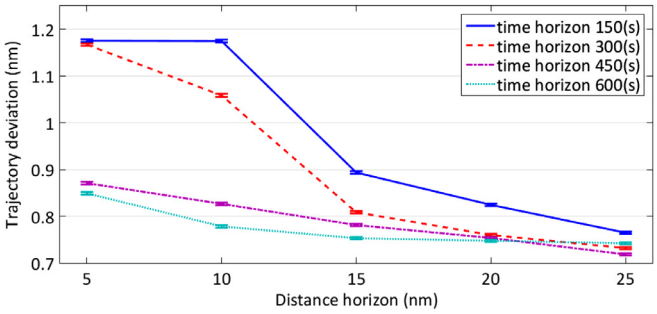
to resolve conflicts, where $\vec{r}_0$ and $\vec{r}_m$ stand for the initial and minimum relative positions between the ego unmanned aircraft and the intruder (Mujumdar & Padhi, 2011).

5.4. Simulation results

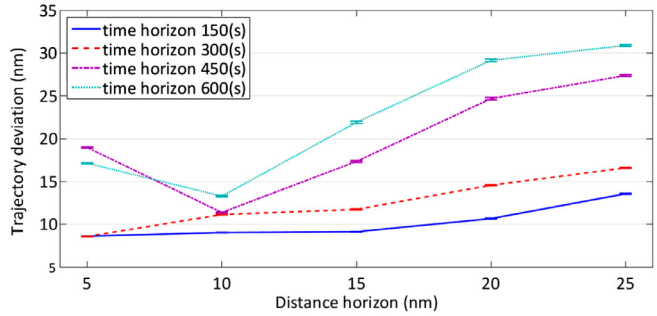
After the necessary pieces explained above are set to create the overall hybrid airspace model, pilot reactions are obtained using the interplay, explained in Section 4, between the level-k game theoretical approach (Section 3.1) and Jaakkola reinforcement learning method (Section 3.2.3). *The algorithm that details this process is given in Algorithm 3.* Once the pilot reaction dynamics are created, quantitative analyses on the integration scenario are conducted using Monte Carlo simulations.

Since one of the bottlenecks of UAS integration into NAS is the maturation of SAA algorithms, we present how the modeling method discussed in this article can be used to conduct comparative quantitative analysis on various aspects of different SAA methods. Specifically, we analyze the effect of *distance horizon* and *time horizon* variables used in the SAA development. In this article, we define the former as the scan radius of the algorithm, and the latter as the amount of projection time used to detect a conflict.

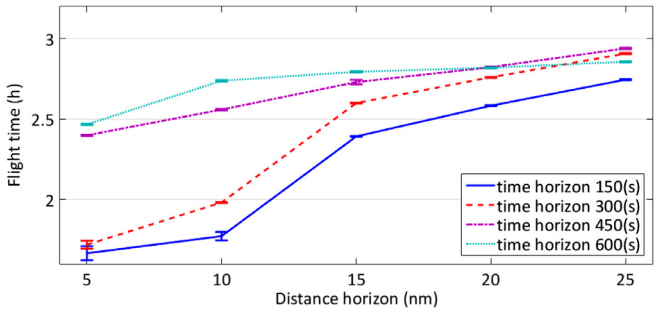
Figs. 7 and 8 depict the effect of varying distance and time horizons on trajectory deviations, flight times and number of separation violations, for SAA1 and SAA2, respectively. A few conclusions, not in increasing or decreasing importance, can be drawn from these results. First, although time horizon makes a significant effect on both safety (separation violations) and performance (trajectory deviations and flight times) measures for SAA1, the unmanned aircraft equipped with SAA2 is not affected as much with the variation of this parameter. Second, the effect of the distance horizon on separation violation numbers levels out quickly for SAA1, while SAA2 keeps showing lesser and lesser violation rates, although SAA1 provides a safer traffic in the same parameter range. Third, UAS trajectory deviations are generally smaller when SAA2 is employed. Fourth, an interesting but expected phenomenon is observed: the trajectory deviations of manned and



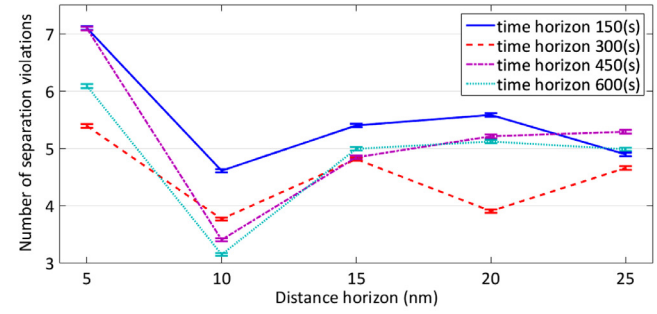
(a) manned aircraft trajectory deviation



(b) UAS trajectory deviation

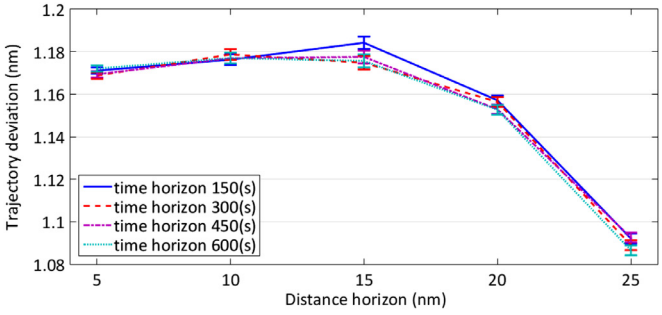


(c) UAS flight time

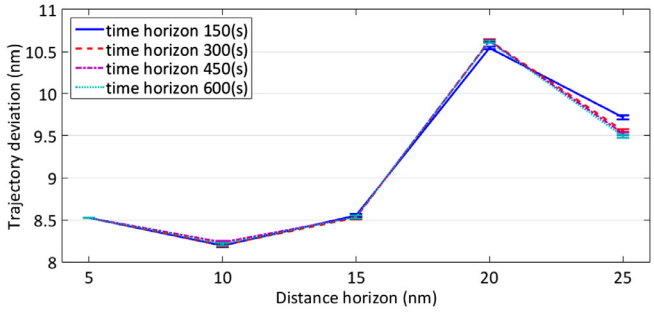


(d) number of separation violations

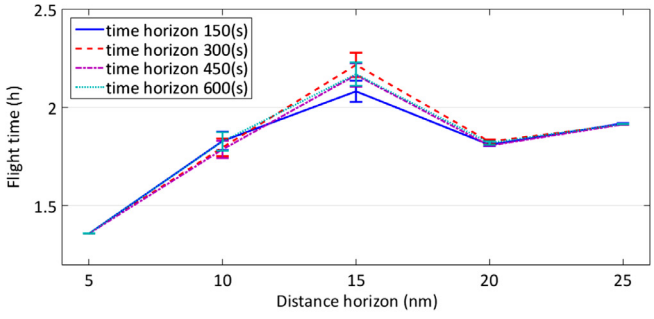
Fig. 7. When SAA1 is employed in HAS, effects of varying time and distance horizons on separation violations, flight times and trajectory deviations are presented. (Musavi et al., 2016, reprinted with permission of the American Institute of Aeronautics and Astronautics, Inc.).



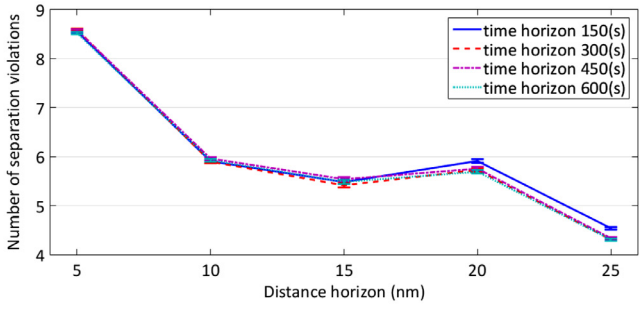
(a) manned aircraft trajectory deviation



(b) UAS trajectory deviation



(c) UAS flight time



(d) number of separation violations

Fig. 8. Safety vs. performance in HAS, when SAA2 is employed is depicted by showing the changes in trajectory deviations, flight times and separation violations with the changes in distance and time horizons. (Musavi et al., 2016, reprinted with permission of the American Institute of Aeronautics and Astronautics, Inc.).

unmanned aircraft have negative correlation. As the UAS trajectory deviations increase, showing that SAA is doing more work for keeping a safe distance, manned aircraft pilots spend less effort and have less trajectory deviations. All these results obtained in a simulation environment with real-time decision making pilots can be used in testing and tuning of SAA algorithms, as well as making a quantitative comparison between different approaches.

5.5. Validation

Although we do not have enough, if at all, UAS integration data to test against the predictions of the game theoretical modeling approach, the resulting models created by the method need to be validated using other means, to achieve at least a minimum level of credibility. Several validation methods are introduced

in the literature, such as *face validity, comparison with validated models, historical data validation, parameter sensitivity analysis* and *predictive validation* (MITRE Corp., 2014). Among these, the ones that are based on data are the most effective validation methods. It is noted that when the data becomes available, one needs to test if relevant statistics between the data and the model are matching. For example, average aircraft trajectory deviation and the average number of separation violations can be compared. Furthermore, during individual encounters, pilot decisions and minimum distances between UAS and manned aircraft, predicted by the model and obtained from the data can be investigated to see whether or not they show similar characteristics. It is noted that the game theoretical modeling approach has enough degrees of freedom, such as reward function terms and weights, to be able to be modified, in case some discrepancies between data and the model are observed when the data becomes available in the future. This is an important feature that needs to be found in predictive models (Law, 2008). Since currently data is not available, we discuss two methods that can be used without data: face validation and comparison with a validated model.

Face validation evaluates two aspects of the model: the logic behind the modeling method, and input–output relationships of the model. Two main approaches used in the proposed model, namely reinforcement learning and game theory, are well-established fields that showed promise in modeling real-life behavior. Experimental backing of the level-k game theory is also provided in relevant references discussed in the review of existing work sections in this article. Also, the logic behind the selection of reward function terms is explained in earlier sections. We also discussed above the input–output relationships when we explained the effects of different parameters on the airspace scenario with figures, and showed that the results are reasonable.

Sample trajectories of a data-validated model developed by the Lincoln Laboratory (Kochenderfer et al., 2008) is available online in two text files that are open to public: `cor_ac1.txt` and `cor_ac2.txt`. A comparison of one of the available trajectories (encounter 3) with the game theoretical model prediction is provided in Fig. 9. In the figure, starting points of aircraft are indicated by thick dots. It is seen that pilot decisions, as well as minimum distance between aircraft during the encounter are similar. Further trajectory comparisons between the game theoretical model and this data-validated model can be found in Musavi et al. (2016).

6. Road traffic modeling

Similar to the unmanned airspace system (UAS) integration study, to obtain the model of the road traffic, we need the physical models of the cars, driver observation and action spaces, and a reward function reflecting the goals and preferences of the drivers. Below, we explain these components and also provide validation of the resultant overall model with traffic data.

6.1. Driver observation and action spaces

Several different traffic scenarios can be modeled via the proposed approach. To be able to test the results with data, we developed the model of a 5-lane highway, similar to the US101 Hollywood Freeway, whose raw data is available at Colyar and Halkias (2007). In this scenario, the human drivers are assumed to be observing (or being able to process from all available data), his or her immediate neighboring lane cars (front left, front right, rear left, rear right) and the car in front. Fig. 10 presents a snapshot of a typical ego vehicle (red) motion, where the driver can observe the surrounding 5 cars. As in the UAS integration scenario, the observation space is quantized and the distances are coded as *nominal, close* and *far*. It is noted that this quantization introduces noise and uncertainty to driver observations since instead of the exact location of neighboring cars, only a certain region occupied by the car is known. The importance of introducing noise and uncertainty to the human observations are discussed in Section 4 and represented by Observation blocks in Fig. 3. To determine reasonable values for quantization, the distance distribution between cars are processed from the raw data provided in Colyar and Halkias (2007) and plotted in Fig. 11. Based on this distribution, *nominal* range is defined to be between 11 m and 27 m, *close* is defined as smaller than 11 m and *far* is used for distances larger than 27 m. It is noted that nominal region consists of approximately half of the area under the curve depicted in the figure. Once the positions of the surrounding cars are defined, their relative motions against the ego car are expressed as *stable, approaching* and *distancing*. As a result, the observation space of the ego car consists of quantized positions and relative motions of the surrounding cars.

As we use traffic data to obtain a meaningful observation space, part of the action space is also formed by considering the acceleration distribution, the plot of which is given in Fig. 12, obtained by processing the same traffic data. According to this acceleration distribution, acceleration inputs of the drivers are categorized into 5 separate continuous sub-distributions: (1) *Maintain*, a normal distribution with zero mean and 0.075 standard deviation, (2) *accelerate* a uniform distribution between 0.5 m/s^2 and 2.5 m/s^2, (3) *decelerate* a uniform distribution between -0.5 m/s^2 and -2.5 m/s^2, (4) *hard accelerate*, a half normal distribution with a 3.5 m/s^2 mean and 0.3 m/s^2 standard deviation, 5) *hard decelerate*, a half normal distribution with -3.5 m/s^2 mean and 0.3 m/s^2 standard deviation. Drivers sample these distributions to create an acceleration action, if they choose to. Two other action choices for the drivers are *move to the left lane* and *move to the right lane*, during both of which the velocity is assumed to remain constant.

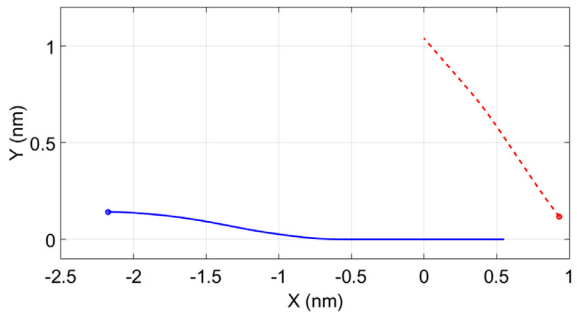
6.2. Reward function

The driver reward function used in the reinforcement learning (RL) algorithm is

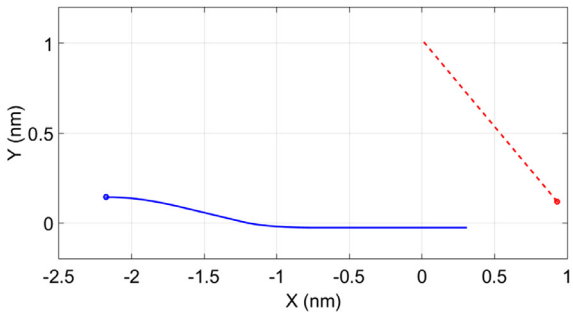
$$r = \omega_1 C + \omega_2 S + \omega_3 D + \omega_4 E, \tag{16}$$

where C is binary, taking values of -1 and 0, depending on whether a collision occurred or not; S is the normalized deviation of the ego vehicle speed from the mean speed of the traffic; D is related to the distance between the ego vehicle and the car in front, and takes the values of -1, 0 or 1 depending on whether the distance is *close, nominal* or *far*, respectively. This term reflects the driver preference of having as much headway as possible. E is the effort variable taking the value of 0 if the action is maintain; a value of -0.25, if the action is accelerate or decelerate; and -0.5 if it is hard accelerate or hard decelerate. E receives -1 if the driver changes lane.

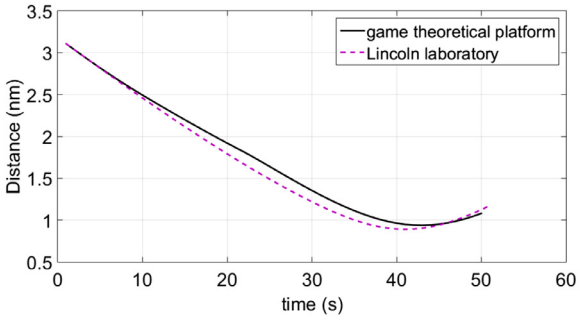
Remark 7. The terms of the reward function provided in (16) are determined based on the specific problems that need to be solved to obtain driver dynamics. The solution of these problems reflects the driver's preferences. Specific issues to be solved can be listed as collision avoidance (term C), performance (term S), safety (term D) and comfort or energy conservation (term E). Instead of discrete valued terms used in (16), continuously varying terms could be employed. However, while this would reflect the driver preferences in a finer manner, it could introduce unnecessary complexity. A more systematic selection of the reward function can be conducted for problems such as imitation learning, where inverse reinforcement learning methods are applied (Sutton & Barto, 2018).



(a) Trajectories created by the validated model.



(b) Trajectories created by the proposed model.



(c) Separation distances for each model.

Fig. 9. Comparison of the trajectories created by the validated model and the game theoretical modeling approach for sample encounter number 3. (Musavi et al., 2016, reprinted with permission of the American Institute of Aeronautics and Astronautics, Inc.).

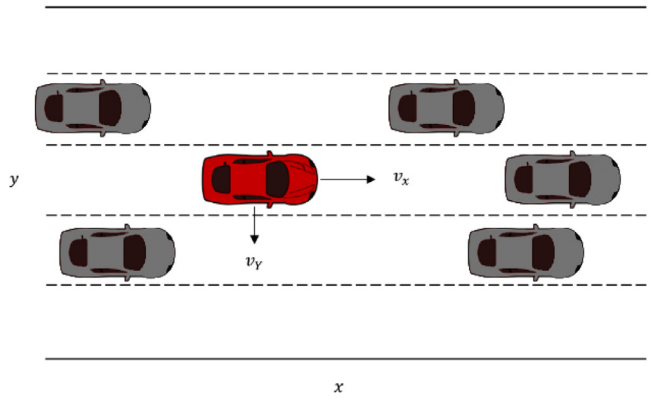


Fig. 10. Ego vehicle and surrounding traffic. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

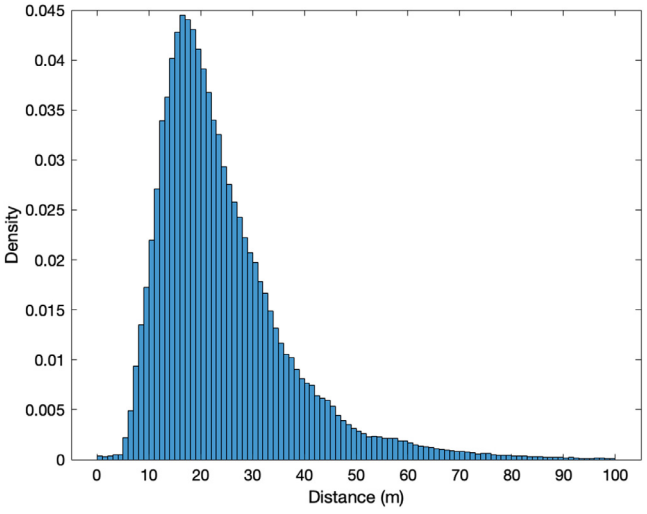


Fig. 11. Distribution of distances to car in front.

6.3. Physical models

For the modeled traffic scenario, simple kinematic models are used to obtain vehicle motion. Once the driver command is received, velocity and position dynamics are obtained as

$$\begin{aligned} x(t+\Delta t) &= x(t) + v_x(t) * \Delta t + \frac{1}{2} a(t) \Delta t^2 \\ y(t+\Delta t) &= y(t) + v_y(t) * \Delta t \\ v_x(t+\Delta t) &= v_x(t) + a(t) * \Delta t, \end{aligned} \tag{17}$$

where v_x and v_y are the velocity components in the x and y directions, x and y coordinates are the same as given in Fig. 10, and Δt is the simulation time step. It is noted that the dynamics are not discrete but continuous, and have to be approximated due to computer implementation.

6.4. Training, average rewards and entropy

Driver policies with various levels are trained using the Jaakkola reinforcement learning algorithm explained in Section 3.2.3. During training, up to 125 vehicles are used in the 5-lane traffic scenario. Figs. 13–15 show the time evolution of average rewards during the training of level-1, level-2 and level-3 policies, together with the average entropy of the probability distribution over actions. The entropy of a probability distribution is calculated as

$$E = -\sum_{i=1}^{n} p_i \log(p_i), \tag{18}$$

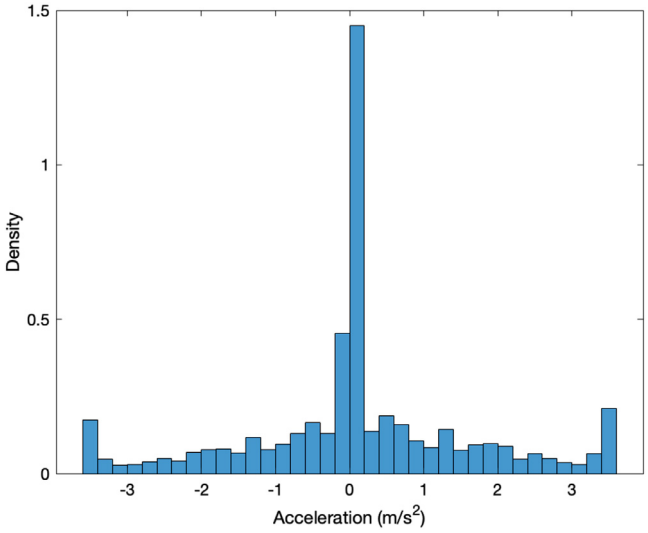


Fig. 12. Acceleration distribution.

where n is the number of probability values p_i. Since the drivers have 7 actions, we have $n = 7$. The entropy is highest when the distribution is uniform, which is the case in the beginning of training, and drops as the training progresses and the distribution gets away from uniform. However, as seen from the figures, although the average reward converges relatively fast, entropy continues to drop at a very slow rate. The reason for this is that there are several states with lower probability of being visited during training and after a certain driving pattern emerges, the effect of these rarely visited states becomes very small. This can also be observed from Fig. 16, where the entropies of the two frequently visited states are shown to converge to very small values.

6.5. Data validation

The driver policies obtained using the game theoretical framework are compared with real traffic data provided by Colyar and Halkias (2007). The data provides acceleration values of the drivers at each measurement instant. This data is first processed to obtain the states the drivers are in and then to find the frequency of taken actions at each state. These state-action frequency distributions form the real driver policies. To compare the distributions modeled using the proposed game theoretical approach and the distributions obtained from data, we use the Kolmogorov–Smirnov (KS) Test for Discontinuous Distributions (Conover, 1972). In this test, if the unknown discrete probability distribution function is $F(x)$ and the hypothesized distribution is $H(x)$, the null hypothesis H_0 is defined as

$$H_0 : F(x) = H(x) \ \ for\ all\ x. \tag{19}$$

Three test statistics used in the test are

$$D = sup_x |H_c(x) - S_n(x)|, \tag{20}$$

$$D^- = sup_x (H_c(x) - S_n(x)) \tag{21}$$

and

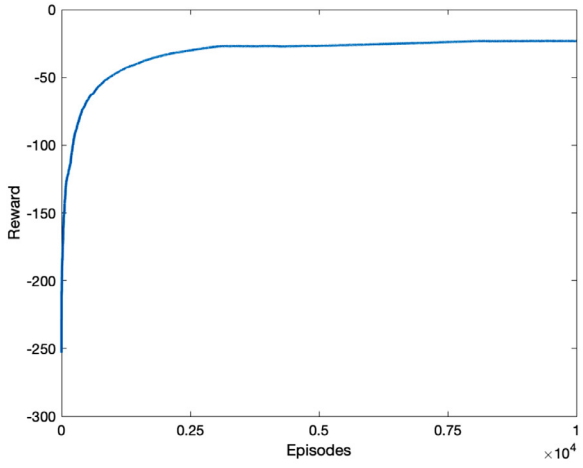
$$D^+ = sup_x (S_n(x) - H_c(x)), \tag{22}$$

where $S_n(x)$ and $H_c(x)$ are the cumulative distribution functions (CDF) of the observed data and of the hypothesized distribution (model), respectively. The observed values of the test statistics D, D^- and D^+ are defined as d, d^- and d^+, respectively. The goal of the KS test is to calculate the probability of observing at least the value d for the test statistics D, which can be stated as $P(D \geq d)$, given that the null hypothesis is true. This is achieved by first calculating $P(D^+ \geq d)$ and $P(D^- \geq d)$, and then obtaining $P(D \geq d)$ as

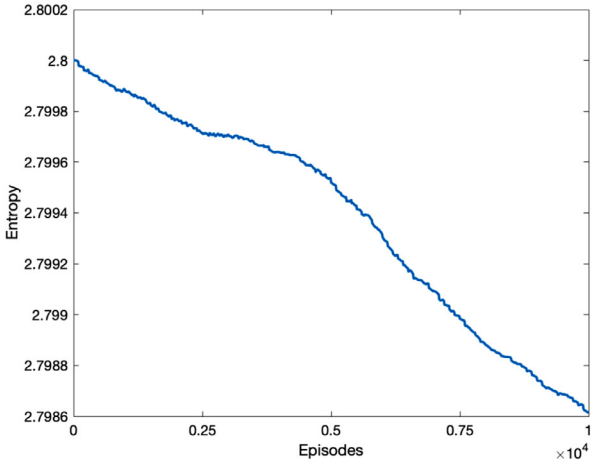
$$P(D \geq d) = P(D^+ \geq d^+) + P(D^- \geq d^-). \tag{23}$$

The details of obtaining $P(D^+ \geq d)$ and $P(D \geq d^-)$ are omitted here for brevity and can be found in (Conover, 1972). Once $P(D \geq d)$ is calculated using (23), the null hypothesis is rejected if $P(D \geq d) \leq 0.05$. Since this test provides meaningful results for distributions with non-zero entries, action probabilities that are lower than 0.01 are set to 0.01 with normalization, for both the real data and the driver model.

Data validation is conducted individually for each driver: for the driver of interest, first, the action probability distributions for each visited state are computed. Second, these distributions are compared with derived driver policies using the proposed modeling framework via KS test. Specifically, the distributions from the data are compared with level-1, level-2 and level-3 policies. Finally, the percentage of the states that can be successfully modeled are reported. This procedure is repeated for every driver whose traffic data is available at Colyar and Halkias (2007). The states that are visited less than a certain threshold number, n_{limit}, during actual

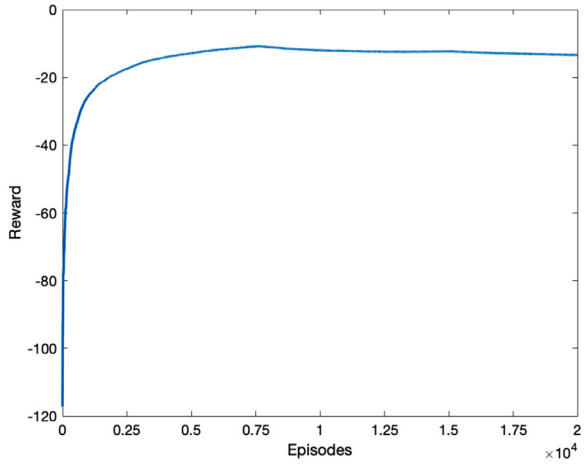


(a) Average reward per episode in level-1 training

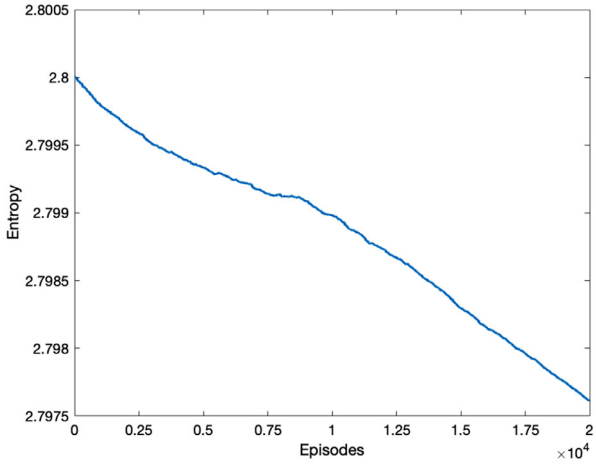


(b) Entropy per episode in level-1 training

Fig. 13. Level-1 training.

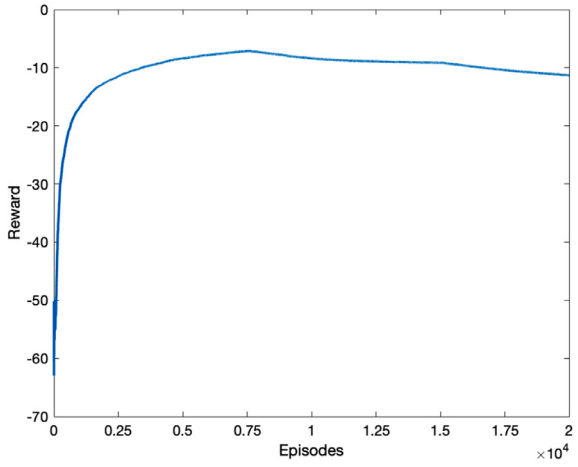


(a) Average reward per episode in level-2 training

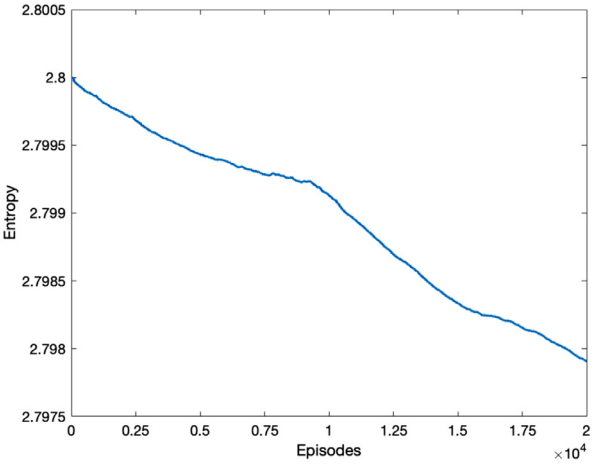


(b) Entropy per episode in level-2 training

Fig. 14. Level-2 training.

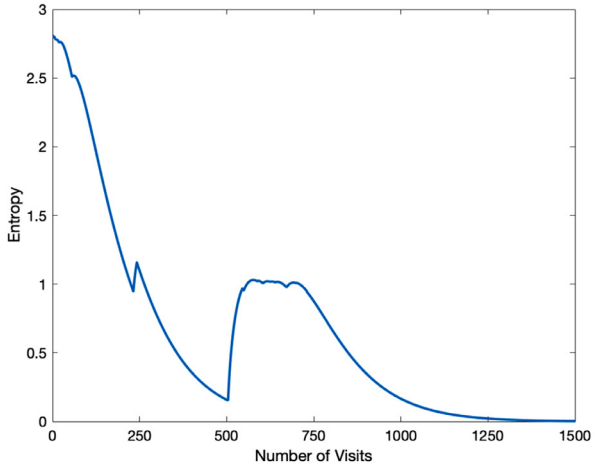


(a) Average reward per episode in level-3 training

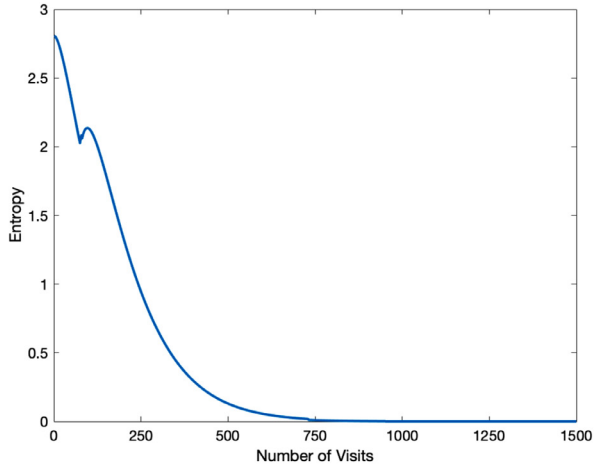


(b) Entropy per episode in level-3 training

Fig. 15. Level-3 training.



(a) Entropy per visit of a state



(b) Entropy per visit of a different state

Fig. 16. Entropy per visit plots of two randomly selected states.

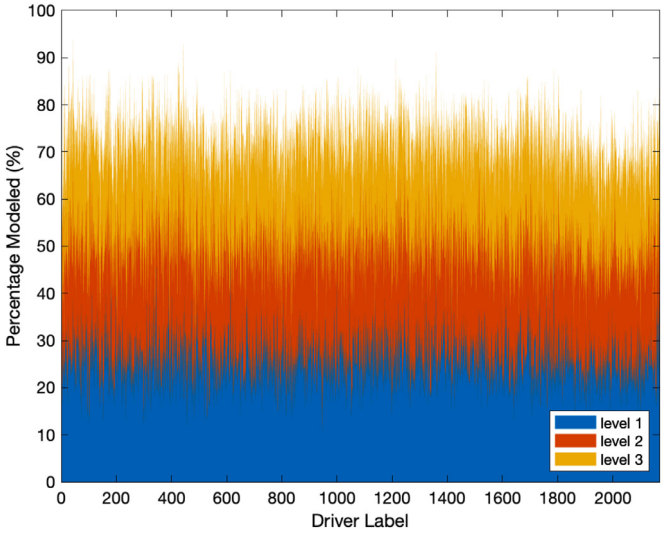


Fig. 17. Percentages of successfully modeled states for each driver, using level-k models, when $n_{limit} = 1$.

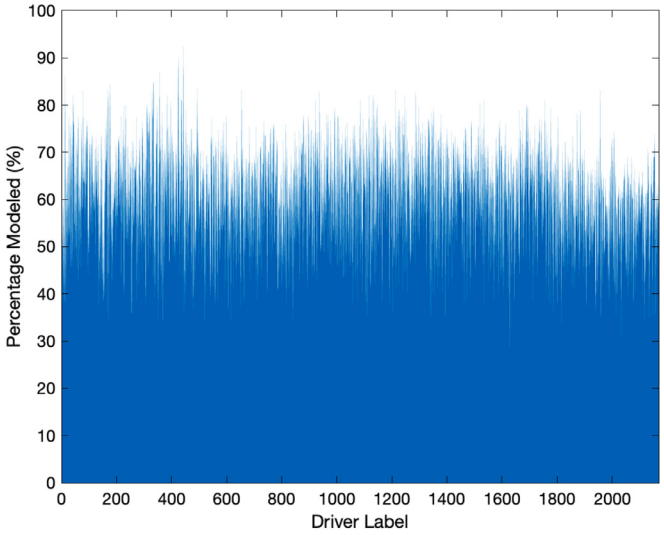


Fig. 18. Percentages of successfully modeled states for each driver, using the uniform distribution model, when $n_{limit} = 1$.

driving by the drivers and during training of the policies are not taken into account. The results are reported for different values of n_{limit}. The algorithm used for this validation procedure is provided in Algorithm 4, where n_{state} is the number of visited states by the driver, $n_{Vdriver}$ is the number of times the driver of interest visited the state being evaluated, n_{Vmodel} is the number of times the state is visited during the training of the driver model, n_{comp} is the number of states that are used in the comparison and $n_{success}$ is the number of states for which the null hypothesis is not rejected.

Data validations using Algorithm 4 are conducted for n_{limit} values of 0, 3 and 5, for each individual driver and the results are reported below.

6.5.1. Validation with $n_{limit} = 1$

Fig. 17 shows the percentage of successfully modeled states for each driver, using level-1, level-2 and level-3 policies as the driver models. Each vertical thin line represents an individual driver and the x-axis shows the driver labels. It is seen that there are more than two thousand drivers in the real traffic data. The y-axis shows

Algorithm 4 Procedure of comparison between one driver-one policy – Kolmogorov Smirnov.

for i = 1 to n_{state} **do**
  if $n_{Vdriver} \geq n_{limit}$ and $n_{Vmodel} \geq n_{limit}$ **then**
    $n_{comp} += 1$
    Set p_i to the probability mass function given by the driver model for the state being evaluated.
    Set k_i to the probability mass function calculated from the driver data for the state being evaluated.
    Set H_c to the cumulative distribution function obtained from p_i.
    Set S_n to the cumulative distribution function obtained from k_i.
    Test the null hypothesis using KS test.
    if Null hypothesis is not rejected **then**
      $n_{success} += 1$
    end if
  end if
end for
Set the percentage of the successfully modeled states to $n_{success}/n_{comp}$.

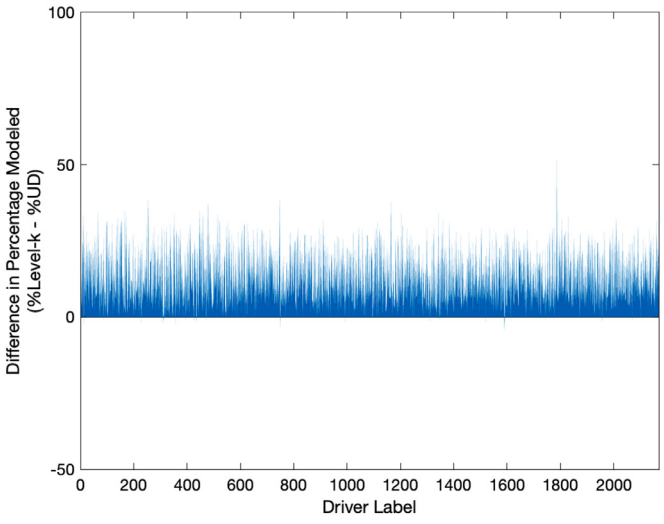


Fig. 19. Difference in successfully modeled state percentages, for each driver, using level-k (combined) and UD models, when $n_{limit} = 1$.

the percentage of successfully modeled states. The colors blue, red and yellow represent the successfully modeled states by level-1, level-2 and level-3 policies, respectively. Fig. 18 also shows the successfully modeled states, but this time using a uniform distribution (UD) model for the drivers. Finally, Fig. 19 shows the difference between the level-k models (combined) and the UD model, in terms of modeling percentages. It is seen that the level-k models cumulatively perform better than the UD model. It is noted that although the level-k models perform better, the percentage of states that are modeled using the UD model are high. The main reason for this is the small n_{limit}, which is 1 in this case. For the state that is visited only 1 time by the actual driver, the KS test does not reject the hypothesis that the action distribution over this state, obtained from the data, is sampled from a UD, since the sample size is not large enough to arrive at a conclusion.

Fig. 20 shows the number of drivers, on the y-axis, and the percentage of the successfully modeled states by the level-k models (top) and the UD model (bottom). For example, it shown that around 80% of the total visited states of 300 drivers are successfully modeled by the level-k models, while only a handful of

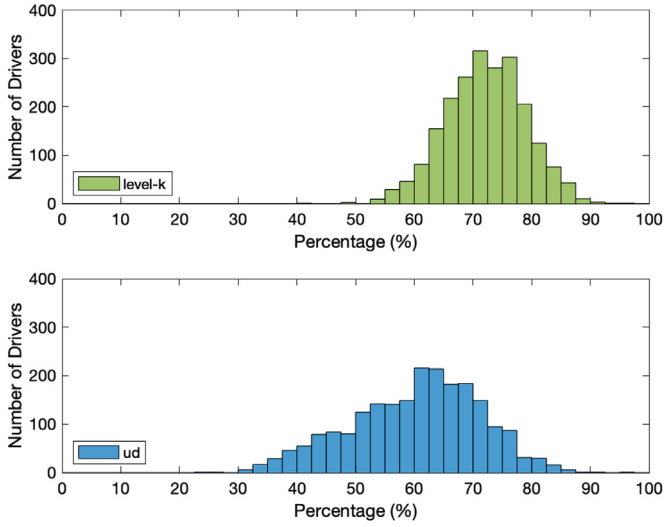


Fig. 20. Distribution of modeled percentages for combined policies and dumb policy, when $n_{limit} = 3$.

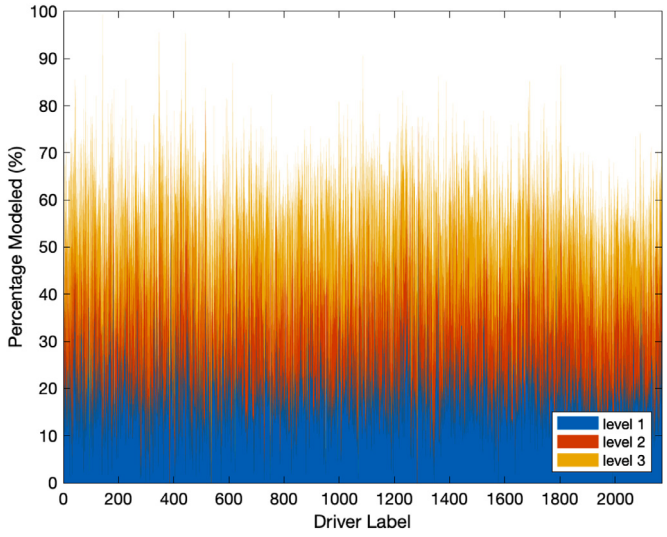


Fig. 21. Percentages of successfully modeled states for each driver, using level-k models, when $n_{limit} = 3$.

drivers' states are successfully modeled up to 60% by the same models. The more the distribution of bars on these figures are grouped on the right, the better, since it shows that successfully modeled state percentages are higher.

6.5.2. Validation with $n_{limit} = 3$

Figs. 21 and 22 show the percentages of the successfully modeled states by the level-k models and the UD model, respectively. As Fig. 23 demonstrates, with an increased threshold ($n_{limit} = 3$) the better performance of level-k models, compared to the UD model, becomes more prominent. This is due to the increased power of the KS test with the elimination of rarely visited states by the drivers.

Fig. 24 also demonstrates the increased performance difference between the level-k and the UD models, in favor of the level-k, by showing the distribution of the number of drivers over successfully modeled state percentages. As more of the rarely visited states are removed from the comparison, the KS test is able to reject the null hypothesis more frequently for the UD models.

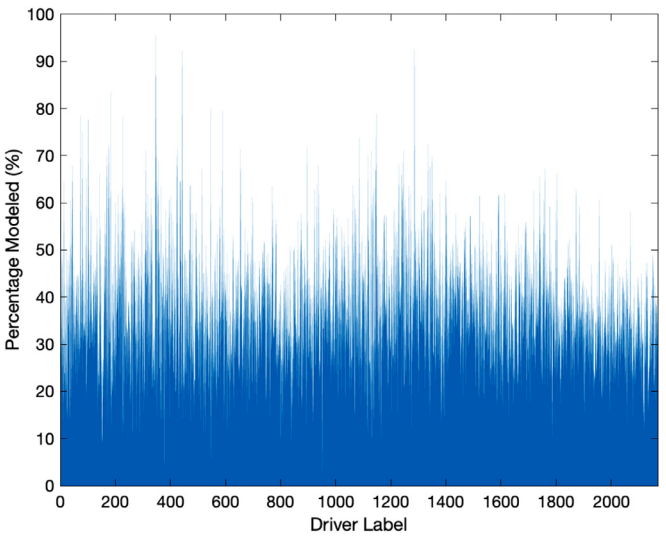


Fig. 22. Percentages of successfully modeled states for each driver, using the uniform distribution model, when $n_{limit} = 3$.

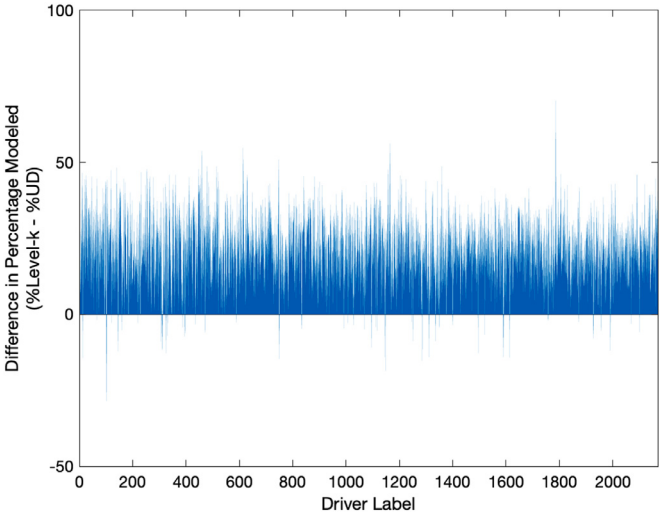


Fig. 23. Difference in successfully modeled state percentages, for each driver, using level-k (combined) and UD models, when $n_{limit} = 3$.

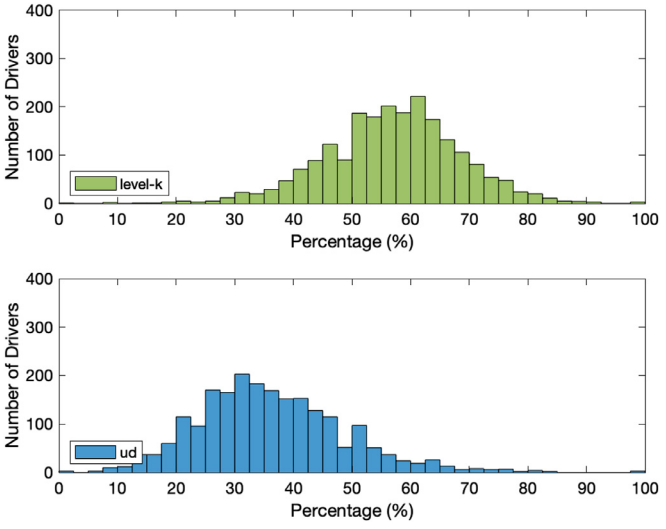


Fig. 24. Distribution of modeled percentages for combined policies and dumb policy, when $n_{limit} = 3$.

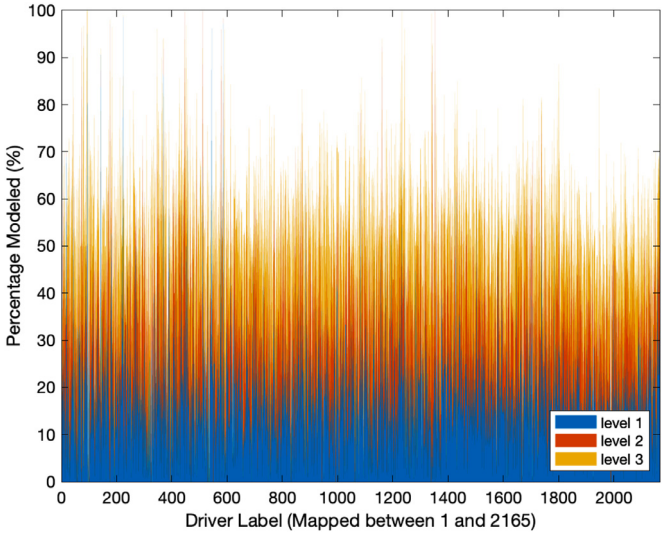


Fig. 25. Percentages of successfully modeled states for each driver, using level-k models, when $n_{limit} = 5$.

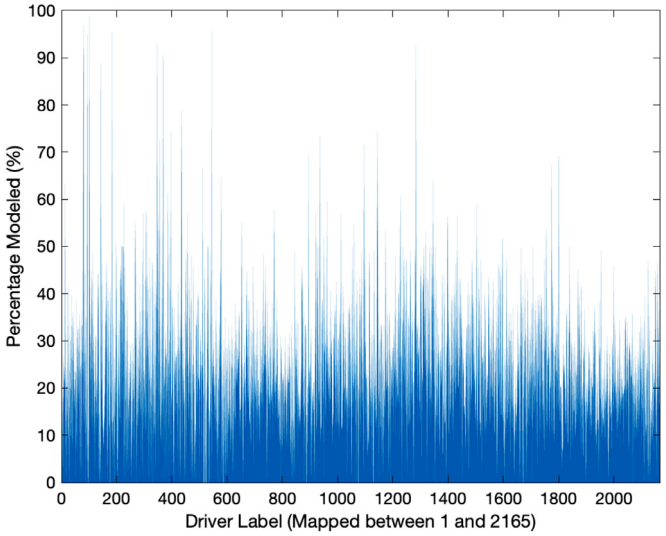


Fig. 26. Percentages of successfully modeled states for each driver, using the uniform distribution model, when $n_{limit} = 5$.

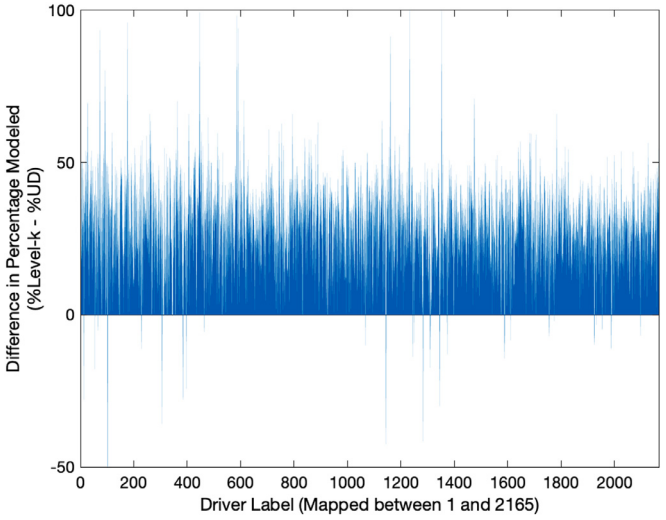


Fig. 27. Difference in successfully modeled state percentages, for each driver, using level-k (combined) and UD models, when $n_{limit} = 5$.

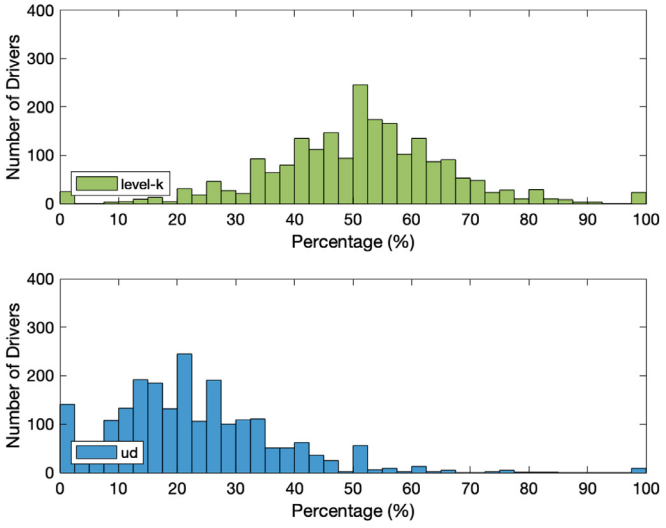


Fig. 28. Distribution of modeled percentages for combined policies and dumb policy, when $n_{limit} = 5$.

6.5.3. Validation with $n_{limit} = 5$

When even more of the rarely visited states are removed from the test by increasing the threshold further to $n_{limit} = 5$, the same trend in increased performance difference between the level-k models and the UD model continues, as seen in Figs. 25–28.

To summarize, KS test results demonstrate that with varying levels of success, the exploited game theoretical modeling framework provides driver models whose predictive power can be validated with real traffic data. Since it is hard to find similar driver models in the literature, with probability distributions over actions, the results are compared with a uniform distribution model. For all three n_{limit} values, level-k policies (combined) performed better than the UD model.

7. Computational complexity

In both of the application areas, hybrid airspace modeling and road traffic modeling, discussed in the previous sections, the exploited game theoretic framework employed Jaakkola reinforcement learning method explained in Section 3.2.3. In this section, a computational cost analysis is provided for this method. Specifically, two questions are answered: (1) How do the action and observation space sizes affect the computational complexity? (2) How do the complexities of the physical motion models affect the computational complexity?

In the analysis, the dimensions of the state and action spaces are taken as S and A, respectively. It is assumed that (a) every state is visited K times during one sweep, and (b) the value functions are updated after each visit. For each state-action pair, (s_i, a_i), the functions given below are calculated:

$$\beta_t(s_i, a_i) = \left(1 - \frac{\chi_t(s_i, a_i)}{K_t(s_i, a_i)}\right)\gamma_t \beta_{t-1}(s_i, a_i) + \frac{\chi_t(s_i, a_i)}{K_t(s_i, a_i)}$$

$$\beta_t(s_i) = \left(1 - \frac{\chi_t(s_i)}{K_t(a_i)}\right)\gamma_t \beta_{t-1}(s_i) + \frac{\chi_t(s_i)}{K_t(s_i)}$$

$$Q_t(s_i, a_i) = \left(1 - \frac{\chi_t(s_i, a_i)}{K_t(s_i, a_i)}\right)Q_{t-1}(s_i, a_i) + \beta_t(s_i, a_i)(R_t - R)$$

$$V_t(s_i) = \left(1 - \frac{\chi_t(s_i)}{K_t(s_i)}\right)V_{t-1}(s_i) + \beta_t(s_i)(R_t - R)$$

$$
\begin{aligned}
&\textit{for each action } a_k \neq a_i \\
&\beta_t(s_i, a_k) = \gamma_t \beta_{t-1}(s_i, a_k) \\
&Q_t(s_i, a_k) = Q_{t-1}(s_i, a_k) + \beta_t(s_i, a_k)(R_t - R) \\
&\textit{for each state } s_j \neq s_i \textit{ and action } a_l \\
&\beta_t(s_j, a_l) = \gamma_t \beta_{t-1}(s_j, a_l) \\
&\beta_t(s_j) = \gamma_t \beta_{t-1}(s_j) \\
&Q_t(s_j, a_l) = Q_{t-1}(s_j, a_l) + \beta_t(s_j, a_l)(R_t - R) \\
&V_t(s_j) = V_{t-1}(s_j) + \beta_t(s_j)(R_t - R) \\
&\textit{for each state } s_x \textit{ and action } a_y \\
&\pi(a_y|s_x) = (1-\epsilon)\pi(a_y|s_x) + \epsilon\pi^1(a_y|s_x). \qquad (24)
\end{aligned}
$$

Therefore, for each state visit, $((24 + (A-1) * (4)) + (S-1) * (A) * (8) + S * A * 4)$ operations are required. Visiting all of the states, K times each, the total number of operations become, $K * S * ((24 + (A-1) * (4)) + (S-1) * (A) * (8) + S * A * 4)$, which can be expressed more compactly as

$$R = c_a S^2 A + c_b SA + c_c S, \qquad (25)$$

where R is the number of required operations, and c_a, c_b and c_c are constants. Therefore, the number of total operations can be given as $\mathcal{O}(|A||S|^2)$.

Using more complex vehicle dynamics, for example increasing the number of differential equations of the vehicle dynamics by c, will result in a computational cost that is c times of the initial cost. Since this a constant effect, the total number of operations can still be expressed by $\mathcal{O}(|A||S|^2)$.

8. Ongoing and future work

In this section, we explain the current and future work about the game theoretical modeling method explained in this paper. Part of these studies are already formulated but not included in this article for a concise and clear exposition. Below, we briefly mention a few that have the potential to improve the existing work in a meaningful manner.

8.1. 3D hybrid airspace

Unmanned aircraft systems (UAS) integration into National Airspace System (NAS) studies presented in this article use a 2-dimensional (2D) geometry for aircraft motion. Although it is demonstrated that the proposed framework can be used to provide significant qualitative analysis power in UAS integration scenarios, the study is still limited and need to be extended to 3D airspace. Preliminary studies are conducted in this direction, details of which can be found in Musavi et al. (2018). One important distinction in this study is the need for an approximate reinforcement learning algorithm to handle the dramatically increased state space due to the 3D geometry.

In the 2D case, the observation space consists of 9 variables. 6 of these variables can take 5 different values, while the remaining 3 variables can take 3 different values. Therefore, the size of the observation space is $5^6 \times 3^3 = 421,875$. This means that 421,875 rows are required in the Q-table to represent each state. To store the values required during training, 16 columns are required. These columns are: state id, state value – V, state visit count, state beta, action 1 probability, action 1 Q value, action 1 count, action 1 beta, action 2 probability, action 2 Q value, action 2 count, action 2 beta, action 3 probability, action 3 Q value, action 3 count and action 3 beta. Hence, $421,875 \times 16 = 6750,000$ values are stored in the Q-table as double. Since each double requires 8 bytes, 54 MB memory is required to store the Q table. This is not a significant amount of memory, which can be handled by any modern computer without any problem. However, when the geometry changes from 2D to 3D, the memory requirement becomes infeasible, even if we keep the action space the same. Adding 6 more observation states with 5 possible values for each, the dimension of the observation space becomes $5^{12} \times 3^3 = 6591,796,875$, which translates into a requirement of 800GB of memory to store the Q table. This calculation shows the necessity to use an approximate reinforcement learning method that eliminates the need for storing a Q table. The Neural Fitted Q-learning method, explained in Section 3.2.2 is currently being tested for this task.

8.2. Large scale cyber-security scenarios

One ongoing study is creating a model of a large scale cyber-attack scenario, where multiple attackers try to hack into a cyber-physical system and several defenders try to keep the system safe. Reliable predictions of attacker-defender dynamics is valuable since they help design systems resilient to cyber-attacks. A two-person model of a cyber-attack scenario of a smart grid system is already conducted by Backhaus et al. (2013). For a larger scenario, similar to the 3D airspace case, problems of increased state space should be solved together with integrating fast optimization algorithms to optimize the system design. Furthermore, reward function design for multiple attackers and defenders is a challenge especially if coordination within the attackers or defenders is envisioned.

8.3. Data validation

Although data validation studies are presented in earlier sections, they are still at their initial stages due to several reasons. First, a lot more data is required for a reliable validation. For example, in UAS integration studies, validation is conducted using a data-validated model of manned aircraft encounters. We hope to obtain UAS and manned aircraft encounter data in the near future as the technology advances. Furthermore, in the traffic scenario, we used US101 data for validation but more road data is expected to be collected to ensure that the proposed model has the capability to model a large variation of highway configurations. In addition, as we use more data, we plan to fit the model parameters to certain amount of data and then validate with independent traffic data. It is noted that parameter fitting to data is not straightforward in the proposed method since reinforcement learning is involved at various reasoning levels. Second, new statistical goodness of fit methods need to be implemented to validate the model's power of prediction. To the best of our knowledge, no goodness of fit methods are implemented for either the UAS integration scenarios or road traffic scenarios where multiple actions are involved. In our ongoing work we are using Chi-square goodness of fit test and Kolmogorov goodness of fit test to validate the method.

9. Open problems and research opportunities

As mentioned in Section 8.1, one of the limitations of the level-k thinking solution concept is that an agent's assumption about other agents' levels does not change during the game, and one solution to this may be the dynamic level-k approach. However, this is only a partial solution since the level types still remain unchanged. For example, if Agent A, after watching Agent B for a few time steps, decides that his or her assumption that Agent B is a level-0 player is wrong, then Agent A needs to update his or her assumption to another level. However, the set of levels he or she can choose from are fixed: Agent A can modify his assumption and assume that Agent B is a level-1 player (instead of level-0), therefore the best response to this is producing the actions of a level-2 player. Agent-A can only change his or her assumption to level-k,

where $k = 1, 2 \ldots, n$, where all levels are already trained and determined. A better, but more computationally expensive approach would be the following: After watching Agent B for a few time steps, Agent A can update his or her assumption by using, for example, a Bayesian update and come up a with a policy that is not in the pre-trained set of levels. After this update, Agent A runs a reinforcement learning algorithm online to determine the best response to Agent B's newly updated policy. To achieve this, we need to find new software and hardware solutions to handle the computational demand.

Another open problem from a control engineering point of view is the problem of stability. Stability in RL is already being studied by the control community as an open problem (Buşoniu, de Bruin, Tolić, Kober, & Palunko, 2018). When used in collaboration with game theory, specifically level-k thinking, the problem becomes even harder to solve, and thus presents a rewarding research direction.

10. Summary

In this article, we reviewed a modeling approach where reinforcement learning and game theory work in tandem to predict cyber-physical human system (CPHS) behavior. Starting from the basic building blocks, we explained the method in detail and then presented two cases where models are created for unmanned aircraft systems (UAS) integration into National Airspace (NAS) and highway traffic scenarios. In both cases, validation studies were discussed using different methods, including using real world data. Finally, we presented ongoing and future works, together with related open problems, which can serve as fruitful research directions.

Declaration of Competing Interest

No potential conflict of interest exists for this article.

Acknowledgements

This effort was sponsored by Turkish Academy of Sciences under the Young Scientist Award Programme.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Available: https://www.tensorflow.org/ [retrieved 30 September 2019].

Albaba, M., Yildiz, Y., Li, N., Kolmanovsky, I., & Girard, A. (2019). Stochastic driver modeling and validation with traffic data. In *Proceedings of the American control conference*. Philadelphia, PA.

Backhaus, S., Bent, R., Bono, J., Lee, R., Tracey, B., Wolpert, D., … Yildiz, Y. (2013). Cyber-physical security: a game theory model of humans interacting over control systems. *IEEE Transactions on Smart Grid, 4*(4), 2320–2327.

Billingsley, T. B. (2006). *Safety analysis of TCAS on Global Hawk using airspace encounter models*. Massachusetts Institute of Technology Ph.D. thesis.

Buşoniu, L., de Bruin, T., Tolić, D., Kober, J., & Palunko, I. (2018). Reinforcement learning for control: performance, stability, and deep approximators. *Annual Reviews in Control, 46*, 8–28.

Camerer, C. F. (2011). *Behavioral game theory: Experiments in strategic interaction*. Princeton University Press.

Carvalho, A., Lefevre, S., Schildbach, G., Kong, J., & Borrelli, F. (2015). Automated driving: the role of forecasts and uncertainty-a control perspective. *European Journal of Control, 24*, 14–32.

Chen, M., Bansal, S., Tanabe, K., & Tomlin, C. J. (2017a). Provably safe and robust drone routing via sequential path planning: A case study in san francisco and the bay area. arXiv: 1705.04585.

Chen, M., Hu, Q., Fisac, J. F., Akametalu, K., Mackin, C., & Tomlin, C. J. (2017b). Reachability-based safety and goal satisfaction of unmanned aerial platoons on air highways. *Journal of Guidance, Control, and Dynamics, 40*(6), 1360–1373.

Chen, M., Hu, Q., Mackin, C., Fisac, J. F., & Tomlin, C. J. (2015). Safe platooning of unmanned aerial vehicles via reachability. In *Proceedings of the 2015 fifty-forth ieee conference on decision and control (CDC)* (pp. 4695–4701). IEEE.

Chen, M., Shih, J. C., & Tomlin, C. J. (2016). Multi-vehicle collision avoidance via Hamilton–Jacobi reachability and mixed integer programming. In *Proceedings of the 2016 IEEE fifty-fifth conference on decision and control (CDC)* (pp. 1695–1700). IEEE.

Colyar, J., & Halkias, J. (2007). US highway 101 dataset. *Technical Report, FHWA-HRT-07-030*. Federal Highway Administration. Available: https://www.fhwa.dot.gov/publications/research/operations/07030/index.cfm[retrieved 9 February 2019]

Conover, W. J. (1972). A Kolmogorov goodness-of-fit test for discontinuous distributions. *Journal of the American Statistical Association, 67*(339), 591–596.

Costa-Gomes, M. A., Crawford, V. P., & Iriberri, N. (2009). Comparing models of strategic thinking in van Huyck, Battalio, and Beil's coordination games. *Journal of the European Economic Association, 7*(2–3), 365–376.

Crawford, V. P. (2008). Modeling behavior in novel strategic situations via level-k thinking. In *Proceedings of the North American economic science association meetings*.

Dalamagkidis, K., Valavanis, K. P., & Piegl, L. A. (2008). On unmanned aircraft systems issues, challenges and operational restrictions preventing integration into the national airspace system. *Progress in Aerospace Sciences, 44*(7), 503–519.

DeGarmo, M. T. (2004). Issues Concerning Integration Of Unmanned Aerial Vehicles In Civil Airspace. *Technical Report*. MITRE Corporation, Center for Advanced Aviation System Development. Available: https://www.mitre.org/sites/default/files/pdf/04_1232.pdf[retrieved 9 February 2019]

Dextreit, C., & Kolmanovsky, I. V. (2014). Game theory controller for hybrid electric vehicles. *IEEE Transactions on Control Systems Technology, 22*(2), 652–663.

Ding, J., Tomlin, C. J., Hook, L. R., & Fuller, J. (2016). Initial designs for an automatic forced landing system for safer inclusion of small unmanned air vehicles into the national airspace. In *Proceedings of the 2016 IEEE/AIAA thirty-fifth digital avionics systems conference (DASC)* (pp. 1–12). IEEE.

Fasano, G., Accardo, D., Moccia, A., Carbone, C., Ciniglio, U., Corraro, F., & Luongo, S. (2008). Multi-sensor-based fully autonomous non-cooperative collision avoidance system for unmanned air vehicles. *Journal of aerospace computing, information, and communication, 5*(10), 338–360.

Florent, M., Schultz, R. R., & Wang, Z. (2010). Unmanned aircraft systems sense and avoid flight testing utilizing ads-b transceiver. In *Proceedings of infotech@ aerospace*.

Fudenberg, D., & Tirole, J. (1991). Game theory, 1991. *Cambridge, Massachusetts, 393*(12), 80.

Gabel, T., Lutz, C., & Riedmiller, M. (2011). Improved neural fitted q iteration applied to a novel computer gaming and learning benchmark. In *Proceedings of the 2011 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)* (pp. 279–286). IEEE.

Hess, R. A., & Modjtahedzadeh, A. (1990). A control theoretic model of driver steering behavior. *IEEE Control Systems Magazine, 10*(5), 3–8.

Hidas, P. (2002). Modelling lane changing and merging in microscopic traffic simulation. *Transportation Research Part C: Emerging Technologies, 10*(5), 351–371.

Jaakkola, T., Satinder, P. S., & Jordan. , I. (1994). Reinforcement learning algorithm for partially observable Markov decision problems. *Advances in Neural Information Processing Systems 7: Proceedings of the 1994 Conference*.

Kacem, T., Wijesekera, D., & Costa, P. (2018). Ads-bsec: a holistic framework to secure ads-b. *IEEE Transactions on Intelligent Vehicles, 3*(4), 511–521.

Kochenderfer, M., Espindle, L., Kuchar, J., & Griffith, J. D. (2008). *Correlated encounter model for cooperative aircraft in the national airspace system version 1.0*. Project Report ATC-344

Kuchar, J., & Drumm, A. C. (2007). The traffic alert and collision avoidance system. *Lincoln Laboratory Journal, 16*(2), 277.

Kuchar, J. K., Andrews, J., Drumm, A., Hall, T., Heinz, V., Thompson, S., & Welch, J. (2004). A safety analysis process for the traffic alert and collision avoidance system (tcas) and see-and-avoid systems on remotely piloted vehicles. In *Proceedings of AIAA third unmanned unlimited technical conference, workshop and exhibit*.

Kumar, P., Perrollaz, M., Lefevre, S., & Laugier, C. (2013). Learning-based approach for online lane change intention prediction. In *Proceeding of the IEEE intelligent vehicles symposium* (pp. 797–802).

Lamnabhi-Lagarrigue, F., Annaswamy, A., Engell, S., Isaksson, A., Khargonekar, P., Murray, R. M., … den Hof, P. V. (2017). Systems & control for the future of humanity, research agenda: current and future roles, impact and grand challenges. *Annual Reviews in Control, 43*, 1–64.

Law, A. M. (2008). How to build valid and credible simulation models. In *Proceedings of the simulation conference, 2008. WSC 2008. winter* (pp. 39–47). IEEE.

Lee, R., & Wolpert, D. (2011). Chapter: Game theoretic modeling of pilot behavior during mid-air encounters. *Intelligent systems reference library series*. Decision making with multiple imperfect decision makers.

Lee, R., Wolpert, D. H., Bono, J., Backhaus, S., Bent, R., & Tracey, B. (2013). Counterfactual reinforcement learning: How to model decision-makers that anticipate the future. In *Decision making and imperfection* (pp. 101–128). Springer.

Lefevre, S., Carvalho, A., & Borrelli, F. (2015). Autonomous car following: A learning-based approach. In *Proceedings of the IEEE intelligent vehicles symposium* (pp. 920–926).

Lefevre, S., Gao, Y., Vasquez, D., Tseng, E., Bajcsy, R., & Borrelli, F. (2014). Lane keeping assistance with learning-based driver model and model predictive control. In *Proceedings of the twelfth international symposium on advanced vehicle control*.

Li, N., Oyler, D., Zhang, M., Yildiz, Y., Girard, A., & Kolmanovsky, I. (2016). Hierarchical reasoning game theory based approach for evaluation and testing of autonomous vehicle control systems. In *Proceedings of the conference on decision and control* (pp. 727–733).

Li, N., Oyler, D. W., Zhang, M., Yildiz, Y., Kolmanovsky, I., & Girard, A. R. (2018). Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems. *IEEE transactions on control systems technology, 26*(5), 1782–1797.

Maki, D., Parry, C., Noth, K., Molinario, M., & Miraflor, R. (2012). Dynamic protection zone alerting and pilot maneuver logic for ground based sense and avoid of unmanned aircraft systems. In *Proceedings of infotech@ aerospace*.

European RPAS Steering Group (2013). Roadmap For The Integration Of Civil RPAS Into The European Aviation System. *Technical Report*. European Commission. Available: http://ec.europa.eu/enterprise/sectors/aerospace/files/rpas-roadmap_en.pdf[retrieved 9 February 2019]

FAA (2013). Integration of Civil Unmanned Aircraft Systems (UAS) in the National Airspace System (NAS) Roadmap. *Technical Report*. U.S. Department of Transportation. Available: http://www.faa.gov/about/initiatives/uas/media/uas_roadmap_2013.pdf[retrieved 9 February 2019]

MITRE Corp. (2014). *Systems Engineering Guide: Verification and Validation of Simulation Models.* http://www.mitre.org/sites/default/files/publications/se-guide-book-interactive.pdf.

Melnyk, R. (2019). A demonstration of reliability and certification standards for unmanned aircraft system control links. In *Proceedings of the AIAA scitech 2019 forum* (p. 1785).

Mujumdar, A., & Padhi, R. (2011). Reactive collision avoidance of using nonlinear geometric and differential geometric guidance. *Journal of Guidance, Control, and Dynamics, 34*(1), 303–311.

Murphy, R., & Shields, J. (2012). The role of autonomy in DoD systems. *Task Force Report*. U.S. Department of Defense, Defense Science Board. Washington, D.C., USA.

Musavi, N., Manzoor, A., & Yildiz, Y. (2018). A 3d game theoretical framework for the evaluation of unmanned aircraft systems airspace integration concepts. arXiv: 1802.07218.

Musavi, N., Onural, D., Gunes, K., & Yildiz, Y. (2016). Unmanned aircraft systems airspace integration: a game theoretical framework for concept evaluations. *Journal of Guidance, Control, and Dynamics, 40*(1), 96–109.

Nancy, A. (2016). *Pilots handbook of aeronautical knowledge*. Washington: US Department of Transportation, Federal Aviation Administration, Flight Standards Service.

Oyler, D., Yildiz, Y., Girard, A., & Kolmanovsky, I. (2016). A game theoretical model of traffic with multiple interacting drivers for use in autonomous vehicle development. In *Proceedings of the American control conference*. Boston, MA.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., … Lerer, A. (2017). Automatic differentiation in PyTorch. In *Proceedings of the NIPS autodiff workshop*.

Perez-Batlle, M., Pastor, E., Royo, P., Prats, X., & Barrado, C. (2012). A taxonomy of uas separation maneuvers and their automated execution. In *Proceedings of the second international conference on application and theory of automation in command and control systems* (pp. 1–11).

Planning, J., et al. (2007). Concept of operations for the next generation air transportation system. *Technical Report*. Citeseer.

Pritchett, A. R. (2010). The system safety perspective. In *Human factors in aviation* (pp. 65–94). Elsevier.

Riedmiller, M. (2005). Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the European conference on machine learning* (pp. 317–328). Springer.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proceedings of the IEEE international conference on neural networks: 1993* (pp. 586–591). San Francisco.

Riedmiller, M., Montemerlo, M., & Dahlkamp, H. (2007). Learning to drive a real car in 20 minutes. In *Frontiers in the convergence of bioscience and information technologies, 2007. FBIT 2007* (pp. 645–650). IEEE.

Salvucci, D., Boer, E., & Liu. , A. (2001). Toward an integrated model of driver behavior in cognitive architecture. *Transportation Research Record: Journal of the Transportation Research Board, 1779*, 9–16.

Salvucci, D. D., & Gray, R. (2004). A two-point visual control model of steering. *Perception, 33*(10), 1233–1248.

Sharp, R. S., Casanova, D., & Symonds, P. (2000). A mathematical model for driver steering control, with design, tuning and performance results. *Vehicle System Dynamics, 33*(5), 289–326.

Sheridan, T. B., Corker, K. M., & Nadler, E. D. (2006). Final report and recommendations for research on human-automation interaction in the Next Generation Air Transportation System. *Technical Report, DOT-VNTSC-NASA-06-05)*. Cambridge, MA, USA: U.S. Department of Transportation, Research and Innovative Technology Administration.

Shia, V., Gao, Y., Vasudevan, R., Campbell, K. D., Lin, T., Borrelli, F., & Bajcsy, R. (2014). Semiautonomous vehicular control using driver modeling. *IEEE Transactions on Intelligent Transportation Systems, 15*(6), 2696–2709.

Stahl, D., & Wilson, P. (1995). On players' models of other players: theory and experimental evidence. *Games and Economic Behavior, 10*(1), 218–254.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge: MIT press.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Treiber, M., Hennecke, A., & Helbing, D. (2000). Congested traffic states in empirical observations and microscopic simulations. *Physical Review E, 62*(2), 1805–1824.

Ungoren, A. Y., & Peng, H. (2005). An adaptive lateral preview driver model. *Vehicle System Dynamics, 43*(4), 245–259.

Vasudevan, R., Shia, V., Gao, Y., Cervera-Navarro, R., Bajcsy, R., & Borrelli, F. (2012). Safe semi-autonomous control with enhanced driver modeling. In *Proceedings of the American control conference* (pp. 2896–2903).

Wakitani, S., Yamauchi, Y., Kinoshita, T., Yamamoto, T., Miyakoshi, M., Harada, S., & Yano, Y. (2018). Design of a vehicle driver model based on database-driven control approach. In *Proceedings of the 2018 IEEE conference on control technology and applications (CCTA)* (pp. 840–845). IEEE.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. King's College, Cambridge Ph.D. thesis.

(2012). In M. Wiering, & M. van Otterlo (Eds.), *Reinforcement learning, state-of-the-art*. Springer.

Wing, D., Prevot, T., Morey, S., Lewis, T., Martin, L., Johnson, S., … Va, H. (2013). Pilot and controller evaluations of separation function allocation in air traffic management. In *Proceedings of the tenth USA/europe air traffic management research and development seminar*.

Yildiz, Y., Agogino, A., & Brat, G. (2013). Predicting pilot behavior in medium scale scenarios using game theory and reinforcement learning. In *Proceedings of the AIAA modeling and simulation technologies (MST) conference* (p. 4908).

Yildiz, Y., Agogino, A., & Brat, G. (2014). Predicting pilot behavior in medium-scale scenarios using game theory and reinforcement learning. *Journal of Guidance, Control, and Dynamics, 37*(4), 1335–1343.

Yildiz, Y., Lee, R., & Brat, G. (2012). Using game theoretic models to predict pilot behavior in nextgen merging and landing scenario. In *Proceedings of the AIAA modeling and simulation technologies conference, AIAA 2012–4487*. Minneapolis, Minnesota.

Yoo, J. H., & Langari, R. (2012). Stackelberg game based model of highway driving. In *Proceedings of the ASME dynamic systems and control conference joint with JSME motion and vibration conference*. Fort Lauderdale, Florida.

Yoo, J. H., & Langari, R. (2013). A Stackelberg game theoretic driver model for merging. In *Proceedings of the ASME dynamic systems and control conference*. Palo Alto, California.