Game Theoretic Modeling of Driver and Vehicle Interactions for Verification and Validation of Autonomous Vehicle Control Systems

Nan Li, Dave W. Oyler, Mengxuan Zhang, Yildiray Yildiz, *Senior Member, IEEE*, Ilya Kolmanovsky, *Fellow, IEEE*, and Anouck R. Girard, *Senior Member, IEEE*

***Abstract*— Autonomous driving has been the subject of increased interest in recent years both in industry and in academia. Serious efforts are being pursued to address legal, technical, and logistical problems and make autonomous cars a viable option for everyday transportation. One significant challenge is the time and effort required for the verification and validation of the decision and control algorithms employed in these vehicles to ensure a safe and comfortable driving experience. Hundreds of thousands of miles of driving tests are required to achieve a well calibrated control system that is capable of operating an autonomous vehicle in an uncertain traffic environment where interactions among multiple drivers and vehicles occur simultaneously. Traffic simulators where these interactions can be modeled and represented with reasonable fidelity can help to decrease the time and effort necessary for the development of the autonomous driving control algorithms by providing a venue where acceptable initial control calibrations can be achieved quickly and safely before actual road tests. In this paper, we present a game theoretic traffic model that can be used to: 1) test and compare various autonomous vehicle decision and control systems and 2) calibrate the parameters of an existing control system. We demonstrate two example case studies, where, in the first case, we test and quantitatively compare two autonomous vehicle control systems in terms of their safety and performance, and, in the second case, we optimize the parameters of an autonomous vehicle control system, utilizing the proposed traffic model and simulation environment.**

***Index Terms*— Autonomous vehicles, game theory, reinforcement learning (RL), traffic modeling, verification and validation (V&V).**

I. Introduction

ONE of the most significant challenges that must be addressed before autonomous cars can be deployed in mass production is the verification and validation of their control systems in terms of safety and performance [1], [2].

Manuscript received August 20, 2016; revised March 13, 2017; accepted June 15, 2017. Date of publication July 26, 2017; date of current version August 6, 2018. Manuscript received in final form June 30, 2017. The work of N. Li and I. Kolmanovsky was supported by the National Science Foundation under Award CNS 1544844 to the University of Michigan. The work of Y. Yildiz was supported by the Scientific and Technological Research Council of Turkey under Grant 114E282 to Bilkent University. The work of A. R. Girard was supported by the Air Force Research Laboratory under Grant FA 8650-07-2-3744 to the University of Michigan. Recommended by Associate Editor K. Butts. *(Corresponding author: Nan Li.)*

N. Li, D. W. Oyler, M. Zhang, I. Kolmanovsky, and A. R. Girard are with the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: nanli@umich.edu).

Y. Yildiz is with the Department of Mechanical Engineering, Bilkent University, Ankara 06800, Turkey.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2017.2723574

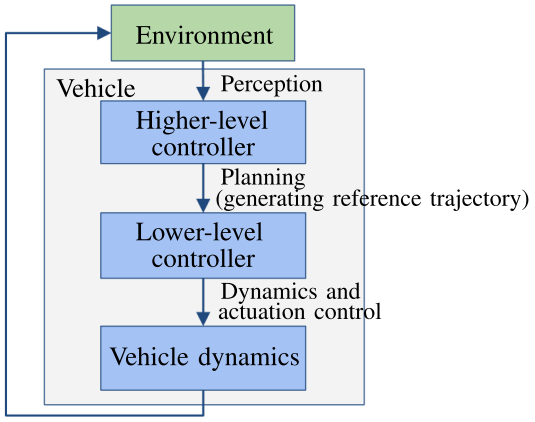


Fig. 1. Control hierarchy of a car.

It has been estimated that autonomous vehicles need to be driven 275 million miles without fatality to assure the same rate of reliability as existing human-driven cars [3]. Hence, testing and calibration of decision and control systems of autonomous vehicles in simulation becomes necessary to complement the on-the-road tests, and the formal method-based and reachability analysis-based procedures (see [4]–[7]).

One common approach to the design of control systems for autonomous vehicles is to utilize a hierarchical control structure (see Fig. 1), wherein a higher level outer-loop controller generates reference trajectories for the lower level inner-loop controller, which, in turn, determines the steering angle and acceleration/deceleration inputs required to track the reference trajectory [8].

Designing these control systems is a challenging task as they need to provide a safe ride together with acceptable performance and comfort in an uncertain traffic environment. Uncertainties generally emanate from uncertain behaviors of other drivers, pedestrians, unexpected obstacles, and changing road and weather conditions. Several control approaches have been proposed for autonomous vehicles, including decision trees [9], [10], partially observable Markov decision processes (POMDPs) [11], and methods based on multipolicy decision-making [12], that are mainly employed as outer-loop controllers. For the inner loop, one of the most common approaches is based on model predictive control [13], [14].

Note that advanced driver behavioral models may also be used in the outer loop [15], [16], with the motivation that an

autonomous vehicle should be able to drive at least as well as a human driver. In fact, some experts have suggested that autonomous vehicles should be permitted on public roads only after it is proven that they are superior to human drivers [2].

Simulators can facilitate the development and testing of autonomous vehicle control algorithms, complementing the road tests. Because autonomous vehicles will be interacting with human-driven vehicles in traffic, high-fidelity simulators need to reflect driver and vehicle interactions. Several methods have been proposed in the literature to address this problem. In [17] and [18], a hidden Markov model-based driver model is proposed based on real driving data. In [19] and [20], *k*-means clustering is used to determine the driving mode and define an approach to predicting and overbounding future vehicle trajectories. It is shown that the performance of an assisted driving algorithm can be improved through the prediction of driver inputs. In [21], a "cognitive architecture" approach, which is "a computational framework that incorporates built-in, well-tested parameters and constraints on cognitive and perceptual-motor processes," is utilized for driver modeling. Built-in logical rules (if-then-else) are used to represent the human decision-making process. In [22], lane change behavior of drivers is modeled using a multiagent simulation system called "simulation of intelligent transport systems." Several logical algorithms are used to model the decision-making during lane changes. The resulting actions of the drivers are, therefore, predefined with strict rules even though driver aggressiveness can be incorporated into the model by tuning certain parameters. In [23], the interactions between autonomous and human-driven vehicles are studied. It is shown by some experimental results that autonomous cars' actions also have effects on what human drivers do. This result further suggests the necessity to reflect the driver-to-driver and vehicle-to-vehicle interactions in a high-fidelity simulator.

With respect to the existing approaches, this paper is distinguished by the advanced modeling of driver and vehicle interactions in traffic using a specific game theoretic formulation that is scalable to multiple vehicles. The proposed method has the following advantages: 1) actions of drivers and vehicles are determined by utilizing a decision-making process, instead of assuming that these actions are prescribed in advance as functions of time or as functions of the state of the system; 2) interactions among multiple human-driven vehicles and autonomous vehicles can be modeled simultaneously and in a computationally tractable way; and 3) all the vehicles in medium-scale traffic scenarios are simultaneously modeled as decision makers as opposed to predicting the decisions of one vehicle while assuming that the rest of the vehicles move based on certain kinematic and dynamic constraints.

In [23], the authors formulate a model that can also reflect the interactions between a human-driven car and an autonomous car, in which both the human-driven car and the autonomous car are modeled as intelligent decision makers. It appears that our work and [23] focus on different aspects and use different approaches. For instance, [23] investigates the interactions between two agents—one human-driven car and one autonomous car—while our work focuses on the interactions among multiple vehicles in traffic in medium-scale scenarios. Furthermore, a simplifying assumption was made in [23] to facilitate solving the formulated two-player game; specifically, the human-driven vehicle model assumes knowledge of the autonomous car control over a short time horizon to compute the best response. In our work, unlike [23], each agent makes decisions separately and simultaneously at each time step, and will not know the decisions of others until their decisions/actions are taken. Finally, the application of a hierarchical reasoning-based game theory is pursued in this paper but not in [23].

Our approach uniquely combines a specific game theoretic formalism—hierarchical reasoning game theory (also referred to as "level-*k*" game theory)—that is used to model intelligent agent interactions, and reinforcement learning (RL), which is used to evolve these interactions in a time-extended (multimove) scenario. The core ideas are synergistic with the framework of "semi network-form games," [24], [25] and help us to obtain the probable outcomes of a complex traffic scenario driven by multiple interactions. To the authors' knowledge, such an approach has not been previously exploited for automotive traffic modeling.

Other game theoretic approaches, in particular, based on Stackelberg games, have been studied for application to vehicle highway driving problems in [15] and [16]. Although these approaches represent driver interactions in traffic using a game theoretic setting, they do not consider dynamic (multimove) scenarios. The latter are considered in [26] for hybrid electric vehicle energy management where the driver and the powertrain are considered to be two players in a game. However, increasing the number of players (drivers, in our case) beyond three complicates computing a Stackelberg solution, especially in a time-extended (multimove) scenario. On the other hand, the hierarchical reasoning-based game theoretic approach exploited in this paper is easily scalable. Indeed, an implementation of the proposed framework for modeling human pilot-to-unmanned aerial vehicle interactions with 50 players and with 180 players can be found in [27] and [28], and scenarios with up to 30 road vehicles are handled in this paper.

The proposed game theoretic model in this paper makes it possible to conduct a quantitative analysis of the traffic. For example, 1) the increase in the number of incidents based on the traffic density can be assessed; 2) various autonomous vehicle control algorithms can be tested and compared quantitatively in a multivehicle time-extended scenario based on certain safety and performance metrics; 3) the effect of a certain parameter value in an autonomous vehicle control algorithm on the safety of the vehicle, e.g., quantified by the number of incidents, can be determined; and 4) optimization of parameter values based on a cost function that reflects safety and performance can be evaluated.

To summarize, the contributions of this paper are as follows.

1) We develop a novel and scalable-to-multiple-vehicles traffic model and simulation environment based on a specific game theoretic modeling framework.
2) Our approach allows the representation of driver interactions for many-move, many-vehicle traffic scenarios using a computationally tractable game theoretic

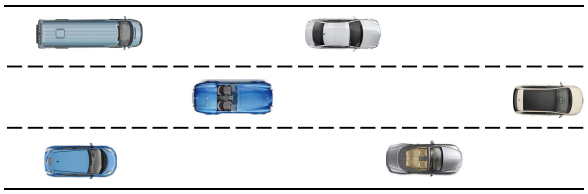


Fig. 2. Example scenario: Traffic in a 3-lane highway.

approach, whereas many existing methods consider only one-move interaction between two vehicles.

3) We demonstrate that autonomous vehicle control algorithms can be quantitatively evaluated in our simulator and compared based on safety and performance metrics applied to simulation outcomes; two policies proposed in the literature serve as case studies, while other autonomous driving algorithms could also be easily integrated and tested.
4) We provide a case study to demonstrate that probabilistic simulation outcomes of traffic scenarios can be utilized to obtain the optimal calibration for an autonomous vehicle controller.

Preliminary results have appeared in [29] and [30]. Differently from [29] and [30], in this paper, 1) we incorporate a more realistic action space including harder brakes and faster accelerations; 2) we develop a more realistic traffic model with more representative distance constraint violation rates via improvements in the RL procedure; and 3) we demonstrate that optimal parameter values for an autonomous vehicle control algorithm can be obtained using a cost function based on safety and performance. Furthermore, this paper has additional details and interpretations that are not found in our preliminary conference papers.

This paper is organized as follows. In Section II, we define the problem being treated in this paper and the vehicle model being used. In Section III, we present level-k game theory and the RL approach to obtain the policies. In Section IV, we introduce two autonomous vehicle control algorithms that will be tested by our simulator. In Section V, we describe our simulator and its implementation results. We then summarize the key developments in this paper in Section VI.

II. Problem Definition

The problem we treat is to model the interactive behavior of drivers in a traffic scenario where the cars are driven on a multilane highway. We later demonstrate that such models can be used in simulators to evaluate autonomous vehicle control policies. Fig. 2 shows an example scenario with three lanes and six cars. Note that this is not a restriction of the proposed method and that scenarios with more cars and lanes can be handled. Simulated cars are assumed to be traveling in the same direction and to be driven by human drivers who obey the general traffic laws.

A. Physical Models

We use the following discrete-time equations of motion (EOM) to model each vehicle during forward motion and lane changes

$$\begin{aligned} x(t+1) &= x(t) + v_x(t)\Delta t \\ v_x(t+1) &= v_x(t) + a(t)\Delta t \\ y(t+1) &= y(t) + v_y(t)\Delta t \end{aligned} \tag{1}$$

where x and y are the vehicle longitudinal and lateral positions, respectively, v_x and v_y are the vehicle longitudinal and lateral velocities, respectively, a is the vehicle longitudinal acceleration, and Δt is the time step size. The longitudinal acceleration $a(t)$ and the lateral velocity $v_y(t)$ are the two control inputs determined by the driver's action decisions. We assume that all cars accelerate and decelerate at either $\pm a_1[\text{m/s}^2]$ or $\pm a_2[\text{m/s}^2]$. The nominal values, $\pm a_1[\text{m/s}^2]$, reflect the acceleration/deceleration a human driver would apply in normal situations, while the hard accelerations/decelerations of $\pm a_2[\text{m/s}^2]$ reflect the values used in urgent situations based on the maximum acceleration/deceleration capability of a vehicle. We also assume that lane changes occur with constant lateral velocity such that the total time to change lanes is t_{cl}[s]. During lane changes, the longitudinal velocity remains constant, and once a lane change begins, it always continues to completion.

The method proposed in this paper can naturally be applied to more complicated models, which may include higher fidelity representations for vehicle dynamics, or incorporate more acceleration/deceleration levels. Although employing a simplified or a more complicated model affects the computational cost, we remark that the computational efficiency of the proposed approach does not originate from some simplifying assumptions on the dynamics, but mainly comes from the hierarchical reasoning game theoretic framework exploited in this paper (see Section III). Within this setting, continuous observation/action spaces can also be handled by an implementation of continuous RL and without an increase in the computational load if appropriate function approximation techniques are applied. Utilizing continuous RL is not pursued in this paper because it has its own issues such as lack of convergence guarantees in certain applications. Its investigation is left to our future work. We note that although the method proposed in this paper can be implemented with more complicated models, the model we use represents a reasonable approximation. For instance, we consider two levels for acceleration/deceleration—one for comfort driving and the other for safety—based on [31] and [32]. Further explanations about the use of discrete observation/action spaces are provided in the following sections.

B. Observation Space

In real traffic flow, a driver can neither observe nor process all the information about all cars on the road. A human can possibly observe and use the information he/she obtains from the cars in a certain vicinity of their own. In particular, a human driver can hardly measure his/her exact distances from other cars. He/she can only estimate the distances and specify them as "close," "far," and so on. The same applies to the relative velocities of other cars. Therefore, we assign the following observation space for the drivers:

1) the longitudinal distance from the ego car to the car directly in front, called range, quantified as "close" (distance $\leq d_c$[m]), "nominal" (d_c[m] $<$ distance $\leq d_f$[m]) or "far" (distance $> d_f$[m]);
2) the range to the car in front and in the left lane, quantified as "close," "nominal," or "far";
3) the range to the car in front and in the right lane, quantified as "close," "nominal," or "far";
4) the range to the car in the rear and in the left lane, quantified as "close", "nominal," or "far";
5) the range to the car in the rear and in the right lane, quantified as "close," "nominal," or "far";
6) the range rate (the time rate of change of the range) to the car in front, quantified as "approaching" (distance decreasing), "stable" (distance not changing), or "moving away" (distance increasing);
7) the range rate to the car in the front left, quantified as "approaching," "stable," or "moving away";
8) the range rate to the car in the front right, quantified as "approaching," "stable," or "moving away";
9) the range rate to the car in the rear left, quantified as "approaching," "stable," or "moving away";
10) the range rate to the car in the rear right, quantified as "approaching," "stable," or "moving away";
11) the lane index of the ego car, quantified as "lane 1," "lane 2," or "lane 3."

Note that this discrete representation of the observation space is based on the "logic-based" human-driver models (see [33]), and it reflects the uncertainty (or noise) present in real-life driver observations. Further discussions on the employment of a discrete observation space can be found in [31] and [32].

Larger observation spaces and finer meshes can be treated in a similar way. In particular, we do not consider the vehicle directly in the back because, in general, it plays a less important role in decision-making when drivers drive on the highway—human drivers usually assume the car directly behind them behaves rationally, that is, it will not rear-end them intentionally. This assumption appears to be adopted by almost all car-following models in the literature (see [33]).

C. Action Space

Drivers are modeled to have seven basic actions:

1) "Maintain" current lane and speed.
2) "Accelerate" at rate $= a_1$[m/s^2], provided velocity does not exceed $v_{\max}$[km/h].
3) "Decelerate" at rate $= -a_1$[m/s^2], provided velocity is above $v_{\min}$[km/h].
4) "Hard accelerate" at rate $= a_2$[m/s^2], provided velocity does not exceed $v_{\max}$[km/h].
5) "Hard decelerate" at rate $= -a_2$[m/s^2], provided velocity is above $v_{\min}$[km/h].
6) Change lane to the left, provided there is a lane on the left.
7) Change lane to the right, provided there is a lane on the right.

This discrete representation of the drivers' action space is based on the "action point" models (see [34], [35]). These seven actions represent a reasonable approximation to the set of human drivers' action decisions in highway traffic [31], [32], although a larger action space can be incorporated without changing the proposed algorithm.

D. Reward Function

A "reward function" is a mathematical representation of the goals of a driver. Basic goals of the drivers in real traffic are: 1) not to have an accident, such as a car crash (safety); 2) to minimize the time needed to reach the destination (performance); 3) to keep a reasonable headway from preceding cars (safety and comfort); and 4) to minimize driving effort (comfort).

These goals can be reflected in the reward function that is given as

$$R = w_1 c + w_2 v + w_3 h + w_4 e \tag{2}$$

where w_i, $i = 1, 2, 3, 4$, is the weight for each term and c, v, h, and e represent "constraint violation," "velocity," "headway," and "effort" metrics. In particular, we define a safe zone for each car (a rectangular area that overbounds the geometric contour of the car with a safety margin) whose boundaries are treated as distance constraints to represent the safety of the car.

The weighting terms, w_i, may change depending on the aggressiveness of the driver, but, intuitively, to avoid distance constraint violation should be of most importance, hence

$$w_1 \gg w_2, w_3, w_4. \tag{3}$$

The terms, c, v, h, e, are explained as follows.

Constraint Violation (c): The term c is assigned a value of -1 when a distance constraint violation occurs and a value of 0 otherwise.

Velocity (v): The term v is assigned the value

$$v = \frac{v_x - v_{\text{nominal}}}{a_1}, \qquad v_{\text{nominal}} = \frac{v_{\min} + v_{\max}}{2}. \tag{4}$$

Here the reason for dividing by a_1 is to make this term of the same order of magnitude as the other terms, to facilitate the choice of weights.

Headway (h): The term h takes the following values depending on the headway distance (the range to the car directly in front):

$$h = \begin{cases} -1 & \text{if headway} \in \text{"close,"} \\ 0 & \text{if headway} \in \text{"nominal,"} \\ 1 & \text{if headway} \in \text{"far."} \end{cases} \tag{5}$$

Effort (e): The term e takes the value 0 if the driver's action is "maintain," $e_2 = -5$ if the driver's action is "hard accelerate" or "hard decelerate," and $e_1 = -1$ otherwise. This term discourages the driver from making unnecessary maneuvers. In particular, a higher penalty discourages the driver from unnecessarily applying "hard accelerate" or "hard decelerate." But in the case where another maneuver cannot avoid a constraint violation, the driver would apply "hard accelerate" or "hard decelerate" to keep safe. Note that the ratio between e_1 and e_2 depends on the driver's behavior and could be tuned to match the driving behavior of different human drivers.

E. Constraints

The reward function (2) already reflects the penalty for distance constraint violations, which may be viewed as imposing soft constraints on the control. For some combinations of states and actions that obviously lead to constraint violations, we can also impose hard constraints to avoid the occurrence of such combinations. In particular, we introduce the following hard constraints, which make certain actions unavailable in certain situations.

1) If a car in the left lane is in a parallel position, the ego car cannot change lane to the left.
2) If a car in the right lane is in a parallel position, the ego car cannot change lane to the right.
3) If a car in the left lane is "close" and "approaching," the ego car cannot change lane to the left.
4) If a car in the right lane is "close" and "approaching," the ego car cannot change lane to the right.

Note that two cars are assumed to be in a parallel position if the safe zones of these two cars intersect in the longitudinal direction. The use of these hard constraints eliminates the clearly undesirable behaviors better than through penalizing them in the reward function, and also increases the learning speed during training.

III. Driver Interaction Model

The driver interaction model developed in this paper enables the modeling of driver-to-driver and driver-to-autonomous vehicle interactions through the use of hierarchical reasoning decision-making and RL. The model is a "policy," which is a stochastic map from the observation space of the driver to his/her action space (see Section II). In other words, this map assigns a probability distribution over possible actions for every observation. In the following sections, we explain how this model is generated.

A. Hierarchical Reasoning Decision-Making

The developed interaction model is premised on the idea that intelligent agents (such as drivers) have different levels of reasoning. A level-0 agent does not consider probable actions or reactions of other agents that he/she is interacting with but rather behaves reflexively. For example, when a driver observes a jam in his/her lane, he/she may make a lane change, trying to bypass the jam, without considering how the other cars would react to this situation. This behavior is referred to as a level-0 behavior and the driver is referred to as a level-0 driver. On the other hand, if a driver assumes that the other drivers are level-0—they would make lane changes as reactions to a jam, such that a heavier jam in the adjacent lane may be caused while the jam in the current lane may vanish as consequences—he/she may therefore decide to stay in the current lane. Then this driver is referred to as a level-1 driver. Similarly, if a driver assumes that the other drivers are level-1 and takes actions accordingly, this driver is a level-2 driver. Higher level driver behavior can be modeled using the same logic. A detailed explanation of this hierarchical reasoning decision-making is given in [36] and [37].

Numerous experimental results from psychology, cognition, and economics have suggested this hierarchical structure in human reasoning (see also [38] and [39]). In [27] and [28], hierarchical reasoning decision-making was exploited to predict pilot behavior, and reasonable predictions were observed. In this paper, we employ this formalism to model driver interactions in traffic, and will show some validations of the developed driver models in Section V.

To obtain higher level policies, one needs to start by defining a level-0 policy. There are various ways to do this, such as selecting each possible action with equal probability regardless of the observation or constructing a very simple policy, which provides a minimally reasonable behavior for a range of observations.

In this paper, a level-0 policy is formulated as follows:

$$\text{action}_{l0} = \begin{cases} \text{"Decelerate,"} & \text{if the car in front is "nomi-} \\ & \text{-nal" and "approaching,"} \\ & \text{or is "close" and "stable,"} \\ \text{"Hard decelerate,"} & \text{if the car in front is "close"} \\ & \text{and "approaching,"} \\ \text{"Maintain,"} & \text{otherwise.} \end{cases} \tag{6}$$

We remark that a driver using this level-0 policy only monitors the motion of the car directly in front without considering the cars in adjacent lanes. If there is a car performing a lane change to the level-0 driver's lane, it becomes a new "front car" once it enters the lane.

B. Reinforcement Learning to Solve the Partially Observable Markov Decision Process

The problem treated in this paper is a multiagent decision-making problem. We use an RL algorithm to determine the policies for each agent based on the reward function defined in Section II-D. To achieve the maximum reward, the RL algorithm exploits two steps including: 1) "policy evaluation," where the state-action pairs are assigned values based on the cumulative rewards they gain and 2) "policy improvement," where the probability of choosing the actions that have higher reward values are increased. For more details on RL see [40].

Conventional RL algorithms require the process to be Markov for convergence guarantees. Note that although the underlying dynamics of the highway problem studied in this paper is Markov, each agent (driver) can observe only a subspace of the whole state space (see Section II-B) and, therefore, has to solve a POMDP problem. In the RL literature, the observations are commonly referred to as "messages." In this paper, we employ the Jaakkola RL algorithm [41], which distinguishes itself from conventional approaches by guaranteeing to converge at least to a local maximum in terms of average rewards, when the problem is of POMDP type. Below, the Jaakkola RL algorithm is summarized. See [41] for further details.

The following steps are followed to obtain the driver policies using the Jaakkola RL algorithm:

Step 1: Evaluate the value function for the messages $V(m|\pi^t)$ and Q-values for the message-action pairs

$Q(m, a|\pi^t)$ associated with the driver policy π^t at step t, using the following equations:

$$\begin{aligned}
\beta_{\mathrm{m}}^t(m) &= \left(1 - \frac{\chi_{\mathrm{m}}^t(m)}{K_{\mathrm{m}}^t(m)}\right)\gamma^{(t)}\beta_{\mathrm{m}}^{t-1}(m) + \frac{\chi_{\mathrm{m}}^t(m)}{K_{\mathrm{m}}^t(m)} \\
V(m|\pi^t) &= \left(1 - \frac{\chi_{\mathrm{m}}^t(m)}{K_{\mathrm{m}}^t(m)}\right)V(m|\pi^{t-1}) \\
&\quad + \beta_{\mathrm{m}}^t(m)[R^t - \bar{R}(\pi^t)] \\
\beta_{\mathrm{a}}^t(m,a) &= \left(1 - \frac{\chi_{\mathrm{a}}^t(m,a)}{K_{\mathrm{a}}^t(m,a)}\right)\gamma^{(t)}\beta_{\mathrm{a}}^{t-1}(m,a) + \frac{\chi_{\mathrm{a}}^t(m,a)}{K_{\mathrm{a}}^t(m,a)} \\
Q(m,a|\pi^t) &= \left(1 - \frac{\chi_{\mathrm{a}}^t(m,a)}{K_{\mathrm{a}}^t(m,a)}\right)Q(m,a|\pi^{t-1}) \\
&\quad + \beta_{\mathrm{a}}^t(m,a)[R^t - \bar{R}(\pi^t)] \qquad (7)
\end{aligned}$$

where m and a designate "message" (the vector containing the 11 observation variables listed in Section II-B) and "action" (one of the seven actions listed in Section II-C), respectively, the superscript t refers to the time step, while the subscript (m or a) indicates whether the function is associated with messages or with message-action pairs. The function χ is an indicator function, taking the value 1 if the message/message-action pair is visited and 0 otherwise. The function K represents the number of times a particular message/message-action pair is visited. The positive sequence $\gamma^{(t)}$ represents a discount factor and is converging to 1 in the limit as $t \to \infty$. The R^t is the one-step reward obtained at the current time instant t, while $\bar{R}(\pi^t)$ is the average reward associated with the policy π^t, i.e., if this policy were executed for an infinite period of time [42].

In the implementation of the above algorithm, the true average reward $\bar{R}(\pi^t)$ is replaced with an estimate, which is computed as an average reward over a past window, making use of the fact that the policy is slowly varying in time.

Step 2: Update the driver policy π, which is a probabilistic mapping from observations (messages) to actions, using the following equation:

$$\pi^{t+1}(a|m) = (1-\varepsilon)\pi^t(a|m) + \varepsilon\hat{\pi}^t(a|m) \qquad (8)$$

where $0 < \varepsilon < 1$ is the learning rate and $\hat{\pi}^t$ is chosen such that $J(m|\hat{\pi}^t) = \max_a \big(Q(m,a|\pi^t) - V(m|\pi^t)\big)$. For any policy $\pi(a|m)$, $J(m|\pi)$ is defined as

$$J(m|\pi) = \sum\nolimits_a \pi(a|m)\big(Q(m,a|\pi) - V(m|\pi)\big). \qquad (9)$$

Note that based on the process defined above, $\hat{\pi}^t$ should be defined in such a way that it has probability 1 for picking the action a that has the highest $Q(m,a|\pi^t)$ value and probability 0 for picking the other actions.

Going through steps 1 and 2 at each time t, the driver model, or the converged policy, denoted by π^*, is obtained once the policy converges during the iterative process. The convergence criterion is based on the convergence of the average reward, i.e., the absolute value of the change in the average reward within a given number of steps being less than a specified tolerance.

C. Role of Hierarchical Reasoning Decision-Making in Obtaining Driver Policies

The process of obtaining driver policies is called "training," where the driver being trained is a *learner* and the other vehicles and automation constitute the *environment*. During the training process, the model of the environment is needed to obtain state transitions as a result of driver actions, where the hierarchical reasoning decision-making approach plays a crucial rule: for the training of a level-k driver policy, all of the traffic but the trained driver are assigned level-(k-1) policies. The process starts with the determination of a level-0 policy (see Section III-A), which represents the lowest level, where the drivers do not consider interactions with other drivers and do not explicitly take into account their possible actions. Once a level-0 policy is determined, the RL algorithm is run by assigning level-0 policies to all of the vehicles except the one that is being trained. At the end of the training process, a level-1 policy is obtained. Similarly, while training a level-2 policy, all of the vehicles but the trained vehicle are assigned level-1 policies. This hierarchical assignment continues until the desired highest level is obtained. In experimental studies [37], it is shown that in human interactions, level-3 players are very rarely encountered and, therefore, in our results we trained policies up to and including level-2.

We remark that the computational efficiency of the proposed framework mainly originates from this hierarchical structure of level-k driver policies. The framework permits the determination of each level-k policy by training a single decision maker, while considering all the other vehicles as part of the environment (because the policies they are using, i.e., level-(k-1), have already been obtained). In this way, two things are simultaneously achieved: 1) the policies are trained in a sequential manner (first the level-0 policy is determined, then the level-1 policy, and so on) and 2) once the level-k policies are determined, the simulations are constructed by incorporating multiple drivers using these policies. Note that in this framework all of the drivers are decision makers and can have different reasoning levels.

IV. Autonomous Driving Control Approaches

The proposed traffic model has been employed to build a simulator to test and evaluate the performance of autonomous driving control algorithms. As specific examples, two autonomous driving approaches, based on Stackelberg policies and decision trees, will be evaluated and compared using a simulator in which the traffic, other than the host vehicle, consists of drivers modeled using our game theoretic policies. In this section, the control algorithms that will be tested are briefly explained and in the next section simulation evaluations are provided.

The Stackelberg policies and the decision tree policies that are compared in this paper were originally developed in [10], [15], and [16]. Since these policies were developed under assumptions representing a simpler traffic environment, some necessary modifications were made to let the autonomous vehicle, which will employ these policies, be able to operate in the traffic environment investigated in this paper.

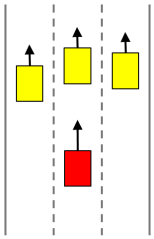


Fig. 3. No solution for the red car by only changing lanes.

For example, the originally proposed Stackelberg and decision tree policies consider only lane change actions. To make them more compatible with the test environment, acceleration/deceleration actions are added to their action spaces. Fig. 3 shows the necessity of this modification, where each rectangle represents a car and the arrows represent both the driving direction and velocity (the longer the arrow, the larger the velocity). In the figure, three yellow cars are in front of the red test car and since the speed of the test car is larger than that of the yellow cars, there is a danger of collision that cannot be resolved only with a lane change.

A. Stackelberg Policies

To obtain the Stackelberg policies for the host vehicle, a path planner is used to choose the actions. At every time step, it considers a game with three vehicles as players – the host vehicle and two other vehicles immediately following it, while the rest of the vehicles are considered to be the environment. The three players are assigned roles as the "leader," the "first follower," and the "second follower," and they choose their actions from the action space in Section II-C sequentially: the leader chooses its action first, followed by the first follower, and, finally, the second follower. Each player evaluates their actions according to a utility function that consists of two parts. The first part, referred to as the positive utility, is defined as follows:

$$U_{\text{pos}} = \begin{cases} \min(d_{\triangle}, d_v), & \text{if there is a vehicle ahead} \\ d_v, & \text{otherwise} \end{cases} \tag{10}$$

where $d_{\triangle}$ is the distance to the car directly in front, i.e., the headway distance, and d_v is the maximum visibility distance. The second part of the utility is referred to as the negative utility and is defined by

$$U_{\text{neg}} = d_{\triangledown} - v_r T - d_{\min} \tag{11}$$

where $d_{\triangledown}$ and v_r are the distance and the relative velocity of the car immediately behind, T is a prediction time window, and $d_{\min}$ is the minimum distance required to allow a lane change; here, $d_{\min}$ is set to the car's safe zone length. Thus, overtaking vehicles are taken into consideration, and lane changes that cut off overtaking vehicles are discouraged.

The actions chosen by the leader, the first follower, and the second follower, denoted by γ_ℓ, γ_{f1}, and γ_{f2}, respectively, are the Stackelberg equilibrium actions, i.e., the leader chooses its actions to maximize its utility for the worst case situation it may face due to the actions chosen by the two followers. Thus, the leader chooses

$$\gamma_\ell^* \in \underset{\gamma_\ell}{\text{argmax}} \min_{\gamma_{f1},\gamma_{f2}} \left[U'_{\text{pos}} + U'_{\text{neg}}\right] \tag{12}$$

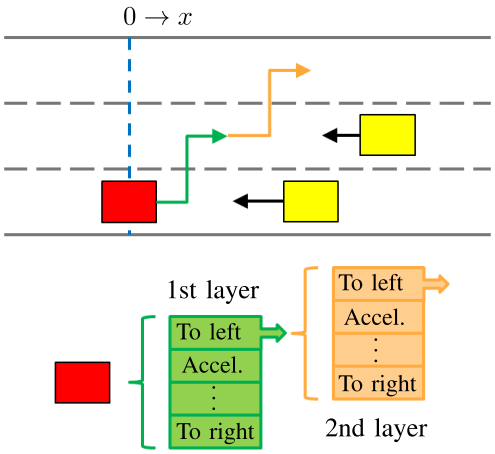


Fig. 4. Decision tree diagram. The black arrows show the relative velocities of the yellow cars with respect to the red test car.

where U'_{pos} and U'_{neg} are the utilities that correspond to a specific set of actions $\{\gamma_\ell, \gamma_{f1}, \gamma_{f2}\}$. The two followers maximize their own utilities with the known choice of γ_ℓ.

In this paper, when constructing the Stackelberg policies, the host vehicle is the "leader" in the game, and the first two cars in its rear (they can be in any lanes) are the followers. An alternative choice in which the host vehicle is one of the followers instead of the leader can be treated similarly. Note that this game lives only in the host vehicle's path planner. The host vehicle will apply the obtained Stackelberg equilibrium action, while the two following cars will choose actions based on their own control policies (in our simulations, the level-k policies), which may be different from their corresponding Stackelberg equilibrium actions.

We remark that in [15] and [16], the considered vehicle dynamics model was different from the one used in this paper. Furthermore, some aspects of the modeling, such as uncertainties in measuring distance, side-viewing, and response delays that were considered in these references are omitted here to simplify the analysis but can be easily integrated.

B. Decision Tree Policies

In the decision tree approach to autonomous driving, the host vehicle's actions are determined by a path planner, which chooses an action for the host vehicle to apply at each time step by evaluating a specified number of preselected action profiles. The path planner builds a tree of potential action sequences (each sequence is referred to as an "action profile"), and evaluates each branch according to a specified metric.

In this paper, the decision tree consists of two layers, and each layer contains the seven actions listed in Section II-C (see Fig. 4), that is, for the evaluation of an action profile, the host vehicle is assumed to apply two actions sequentially, one per layer, each of which is chosen from the seven actions described in Section II-C. Thus, each action profile consists of two actions, and therefore, $7^2 = 49$ profiles are evaluated to determine the optimal one. The evaluation metric is based on the reward function (2), which is also used for the training of the level-k policies. In particular, the total reward is calculated as a

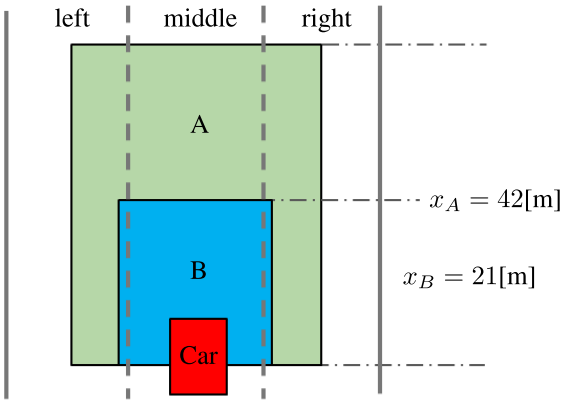


Fig. 5. Triggering threshold.

weighted sum of the rewards obtained from the two layers

$$R_{\text{total}} = w_{l1} R_{l1} + w_{l2} R_{l2} \tag{13}$$

where $w_{l1}, w_{l2} \in \mathbb{R}^+$ are the weighting terms, and R_{l1} and R_{l2} are the reward functions for layer 1 and layer 2, which are the same as (2). After the evaluations, the host vehicle applies the first-layer action of the profile that has the highest total reward among all profiles. This procedure is repeated at every step. When evaluating the action profiles, the host vehicle assumes that all other vehicles would apply the action "maintain" over its prediction horizon.

In [10], it was assumed that the environment evolves deterministically, independently of the host vehicle's actions. However, in our simulator, not only does the host vehicle respond to the traffic, but the traffic also responds to the host vehicle's actions using the level-k policies.

C. Path Planner Triggering Threshold

Both the Stackelberg and the decision tree policies are used as path planners and triggered only when necessary and beneficial. When the policies are not triggered, the vehicle follows a predefined driving pattern. More specifically, referring to Fig. 5, the driving algorithm can be explained as follows:

1) "Accelerate," if there are no other cars in region A.
2) Path planner is triggered, if there are cars in region A, but no cars in region B.
3) "Safe mode," which in this paper is the same as the level-0 policy, is applied if there are cars in region B.

It is clear that if there are no cars in region A, the host vehicle may accelerate safely, while if there are cars in region B, the host vehicle should decelerate to keep a reasonable headway distance. The activation logic is designed: 1) to increase the safety and 2) to reduce the computational cost by preventing unnecessary action evaluations. In particular, region A is designed to cover the center of adjacent lanes, while region B is designed to cover the boundary lines of the current lane, so that when some vehicle in the vicinity is changing lanes into the host vehicle's lane, the host vehicle becomes aware of it before the other vehicle actually enters its lane.

Note that the geometric parameters of these regions can be optimized. The proposed simulator can help to calibrate these parameters and find their optimal values. In Section V, we will present one case study of using our simulator to optimize the parameter values of given autonomous driving policies.

V. Results

A. Environment and Setup

We model the environment as follows. The width of a lane is 3.6 m, and the safe zones around the cars, which should not be violated, are modeled as 6 m $\times$ 2 m boxes. Cars always drive at the center of a lane unless they are changing lanes. Cars only accelerate or decelerate at $\pm a_1 = \pm 2.5$ m/s^2 or $\pm a_2 = \pm 5$ m/s^2, and lane changes occur with constant lateral velocity such that the total time to change lanes is $t_{cl} = 2$ s. During lane changes, the longitudinal velocity remains constant and once a lane change begins, it always continues to completion. The longitudinal axis is called x, and its origin is collocated with the car that is to be trained or evaluated.

To configure the simulation, the following values need to be specified:

1) the number of lanes, n_ℓ;
2) the number of cars, n_c;
3) the maximum allowable initialization distance, $x^0_{\max}$;
4) the simulation duration, t_f.

The following procedure is employed to initialize a simulation. A car is assigned to a lane that is determined randomly based on the uniform distribution. The specific location of the car within the assigned lane is determined randomly based on the uniform distribution in $[-x^0_{\max}, x^0_{\max}]$. Then, the initial longitudinal velocity of the car is given randomly based on the uniform distribution within the range $[v_{\min}, v_{\max}] = [62, 98]$ km/h. The initial action is set as "maintain." The car is then assigned a policy to follow (level-0, 1, or 2). This process is repeated until all cars are configured. While locating each car, it is required that the minimum initial distance between every pair of cars in the same lane be 30 m. Once the initialization is completed, the simulation is run according to Algorithm 1 as given below.

Five cars are observable by the ego car, as described in Section II-B, and a car is considered "close" if its relative longitudinal position satisfies $|x_r| \le d_c = 21$ m, "nominal" if $d_c < |x_r| \le d_f = 42$ m, and "far" if $d_f < |x_r| \le d_v = 63$ m, where d_v is the maximum visibility distance. Cars farther away than d_v [m] are considered to be out of visual range and unobservable. If no car can be observed in a position, this is considered equivalent to a car that is "far" and "moving away." Note that $d_c = 21$ m is determined by considering the minimum distance needed for a car to safely avoid a distance constraint violation by braking with the maximum allowable deceleration in the worst case scenario where a car in front is approaching with maximum relative speed and entering the "close" range.

Fig. 6 shows a snapshot of an example simulation setup with three lanes. The rectangles represent the safe zones around the cars, and the arrows attached to the rectangles show the relative velocities of the cars with respect to the ego car, that is located in the center lane at $x = 0$. Specific observations by the ego car are given as follows:

Algorithm 1 Single Episode Simulation

```
t = 0.
while t < t_f do
    foreach car do
        Obtain observations from the environment.
        Given the observations, determine an action
          based on assigned policy.
        Given the action, update position and velocity.
    end
    if training a policy then
        Evaluate reward function for trainee.
        Update value function.
    end
    if trainee/test vehicle is in a constraint violation state
      then
        End the simulation.
    end
    t = t + Δt.
end
```

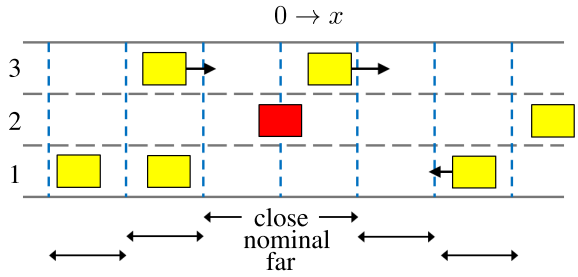


Fig. 6. Simulation environment.

1) front left: close, moving away;
2) front center: far, moving away;
3) front right: far, approaching;
4) rear left: nominal, approaching;
5) rear right: nominal, stable;
6) lane index: lane 2.

Note that two cars in Fig. 6 are unobservable: The car in the front center position is beyond the visual range—so the corresponding observed range rate status is "moving away" even though the car is actually "stable"; and the car in the rear right "far" position is hidden by the car in the rear right "nominal" position. These limitations in the observation space reflect the POMDP nature of the problem, as discussed previously.

B. Level-0 Driver Behavior

In this section, we present simulation results to show the driving behavior of a level-0 car. Furthermore, we present the simulator user interface.

In Fig. 7, the red car in the center is the trainee/test vehicle, while the yellow cars make up the traffic environment. The red arrow in front of the red car indicates its travel direction, and arrow length indicates how fast the car is traveling. The panel on the left is a speedometer, and the steering wheel on the right indicates the lateral motion of the car. The green box

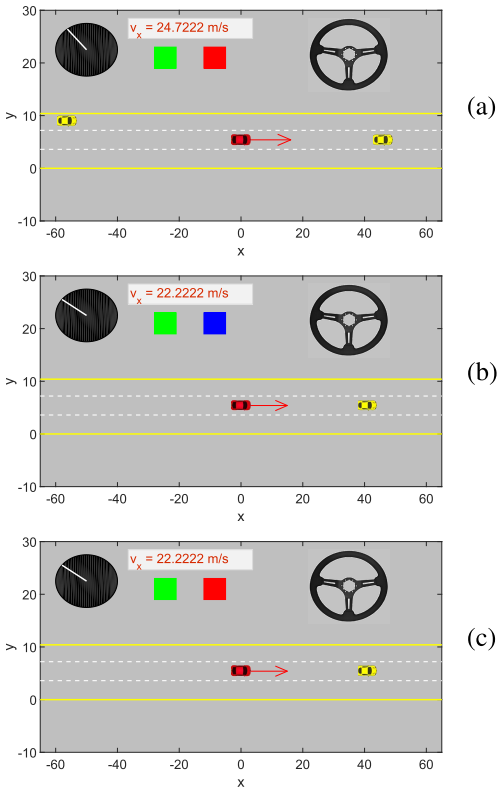


Fig. 7. Level-0 simulation results. Snapshots of the simulation at (a) 40 s, (b) 42 s, and (c) 44 s, respectively.

and red box in the middle indicate the gas pedal and the brake pedal, respectively, and turning blue indicates that the pedal is pressed. The coordinate axis is fixed on the test car and the motions of the other cars can be tracked by their relative distances to the red car. In Fig. 7(a), a yellow car directly in front of the red car ("far") is "approaching" because the red car is faster. At this moment, neither the gas pedal nor the brake pedal is pressed. In Fig. 7(b), the yellow car enters the "nominal" range, and consistently with the level-0 policy, the red car brakes and decreases its speed. In Fig. 7(c), the red car gets to a lower speed, and the yellow car is now "stable."

C. Training Process

When training a new policy, the observation value function, V, for observed message m, and the action value function, Q, for message-action pair (m, a), are initialized as follows:

$$\begin{aligned} \forall m, \quad & V(m) = 0 \\ \forall m, \quad \forall a, \quad & Q(m, a) = 0. \end{aligned} \tag{14}$$

For each observation, the actions are assigned equal probability of selection at initialization, and during each policy improvement step, if

$$\max_a Q(m, a) > V(m) \tag{15}$$

then 0.01 is added to the probability of selecting $\text{argmax}_a\, Q(m, a)$, after which the action probabilities are normalized. We remark that this procedure is one way of realizing (8) that corresponds to using a time-varying ε. An alternative way is to use a constant ε at each step.

Algorithm 2 Training Process

episode=0;
while *episode* $<$ *maximum training episodes* **do**
  Randomly select $n_c \in [0, n_c^{\max}]$.
  Initialize the trainee with level-k policy.
  Initialize all cars other than the trainee with level-(k-1) policies.
  Evaluate the level-k policy using Algorithm 1.
  Improve the level-k policy.
  episode=episode+1.
end
Assign level-0 policy to the messages not visited enough.

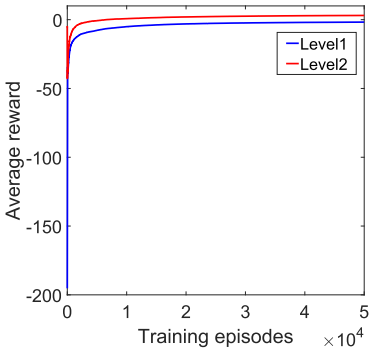


Fig. 8. Evolution of the estimated average reward during level-1 and level-2 training.

The observation space described in Section II-B has 3^{11} different observations. In order to ensure that the learning algorithm is exposed to a large portion of the observation space, the trainee needs to be exposed to a range of traffic environments. Therefore, during training, the number of cars in the environment is selected randomly, based on the uniform distribution, where $0 \leq n_c \leq n_c^{\max}$. The maximum number of cars, $n_c^{\max}$, is chosen based on the number of lanes and $x_{\max}^0$, such that if $n_c^{\max}$ cars are placed in the environment, the road is near full capacity.

Finally, after sufficient training time, the level-0 policy is assigned to the observations (messages) that are still not visited enough during training, so that conservative actions are performed in such rarely encountered cases.

Training then proceeds according to Algorithm 2, as given above:

Fig. 8 shows the evolution of the estimated average reward during level-1 and level-2 training. As can be observed from the figure, the algorithm does converge in both cases. The reward function weights are chosen as: $w_1 = 10\,000$, $w_2 = 5$, $w_3 = w_4 = 1$. These weights can be tuned manually or calibrated using traffic data through approaches such as inverse optimal control [43] or inverse RL [44]. This is left to the future work.

D. Level-k Driver Interactions

We first present simulation results to show the driving behavior of a level-1 car in a level-0 traffic environment (Fig. 9), and then of a level-2 car in a level-1 traffic environment (Fig. 10). It is noted that in both figures, the traffic

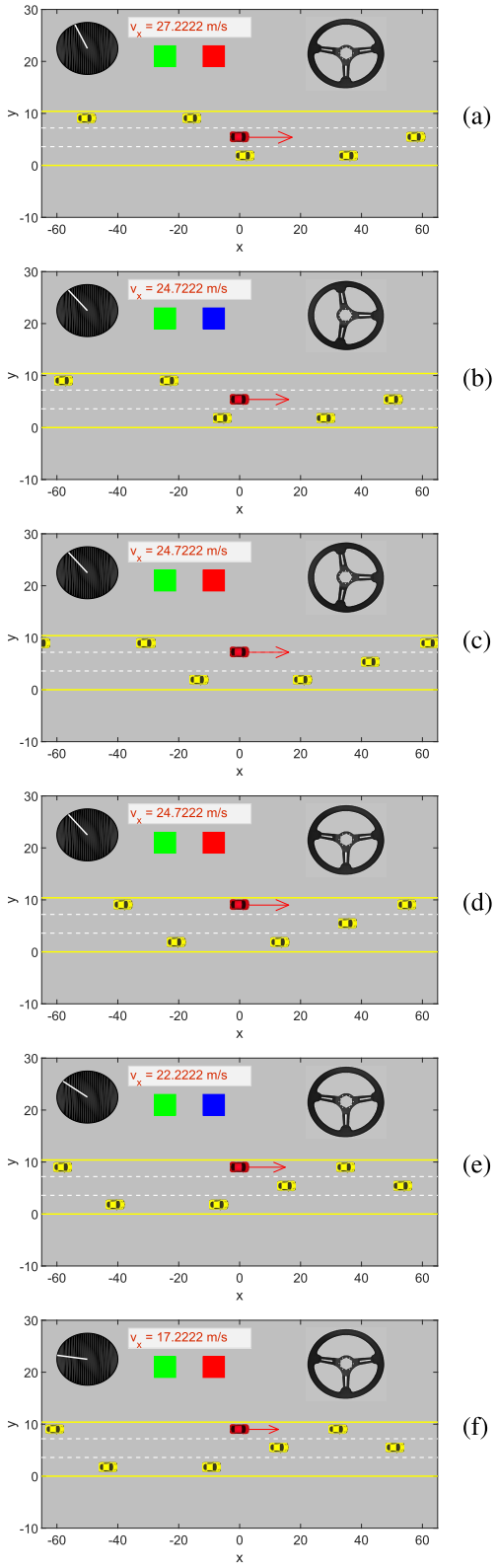


Fig. 9. Level-1 versus level-0 simulation results. Snapshots of the simulation at (a) 178 s, (b) 179 s, (c) 180 s, (d) 181 s, (e) 184 s, and (f) 188 s, respectively.

is moving toward the right. The test vehicle is red and the rest of the vehicles are yellow.

In Fig. 9(a), the yellow car directly in front of the red car is "approaching" with a large relative speed and enters the

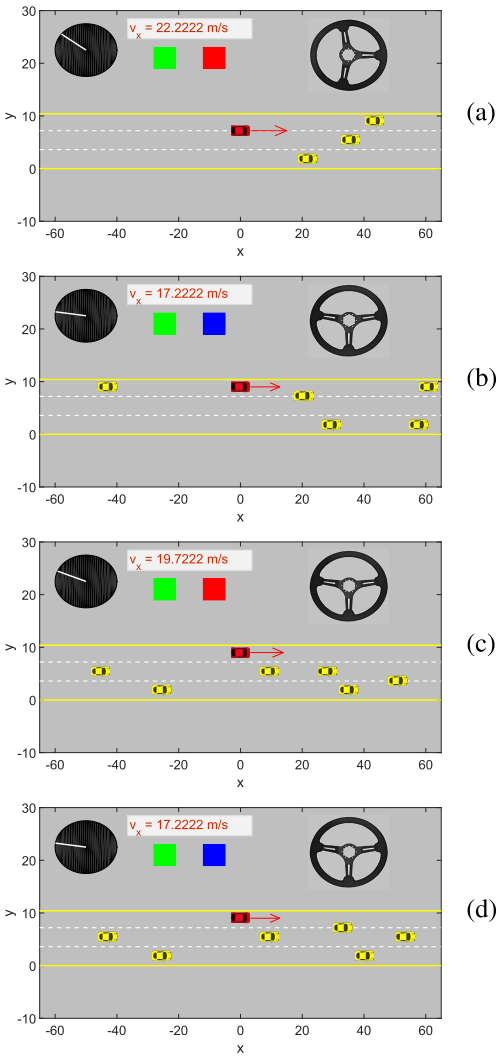


Fig. 10. Level-2 versus level-1 simulation results. Snapshots of the simulation at (a) 60 s, (b) 66 s, (c) 79 s, and (d) 80 s, respectively.

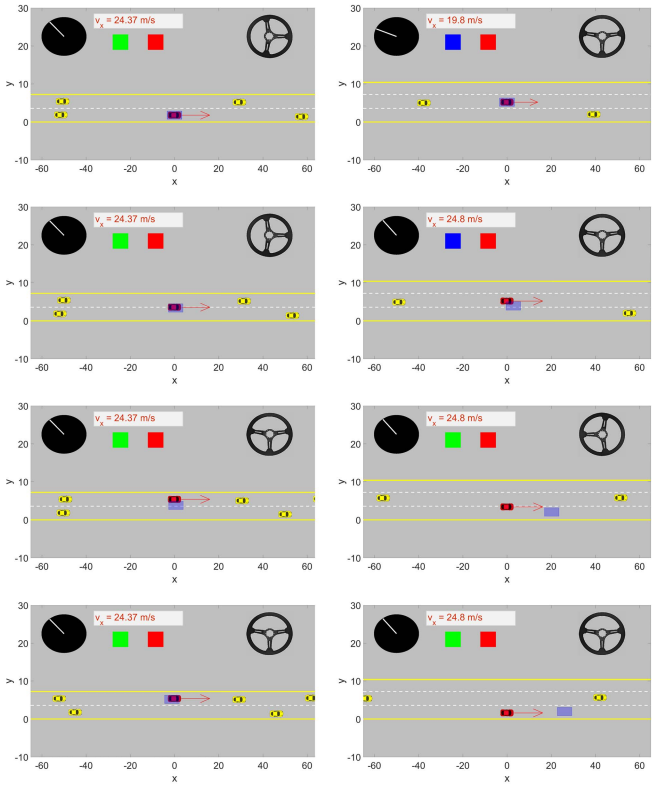


Fig. 11. Validation of level-1 (left column) and level-2 (right column) driver models. The red car uses a level-k policy, and the blue box represents the behavior of a real driver.

"far" distance range. In Fig. 9(b), the red car decelerates and begins to steer. In Fig. 9(c), the red car starts to move into the lane to its left. In Fig. 9(d), the lane change is completed, after 2 s, while another yellow car in front approaches the red car. In Fig. 9(e), the red car decelerates until it keeps a stable headway, which is shown in Fig. 9(f). All these actions represent a reasonable driving behavior.

Similarly, Fig. 10 shows a simulation result of a level-2 car in a level-1 traffic environment. In Fig. 10(a), the yellow car in the middle lane is "approaching" the red car, while the yellow car in the left lane is "moving away," so the red car decides to change lane to the left. In Fig. 10(b), the yellow car in the middle lane starts to change lane to the left (because of some cars in its front, which are not shown in this figure), so the red car needs to brake. In Fig. 10(c), there is no car directly in front of the red car, in which case a level-1 car would accelerate. However, because the longitudinal distance of the yellow car in the middle lane to the red car is too small and the red car has no confidence that the yellow car will not change lane to the left, in which case the red car's acceleration would lead to a distance constraint violation, so the red car decides to maintain its current speed. In Fig. 10(d), there is a car in the traffic moving into the red car's lane, which forces the red car to decelerate.

Different driving patterns of level-1 and level-2 cars, in similar situations, reflect different levels of reasoning in decision-making. It is noted that in Fig. 10, the cars in the environment (yellow) also change lanes, unlike the ones in Fig. 9, because their actions are based on the level-1 policy, which allows lane changes.

To validate our level-k driver models, we compare the model behavior with the human driver behavior in similar decision-making situations using real traffic data. Specifically, we replace one car in traffic flow data by a car controlled by our level-k policy and compare the level-k car future trajectory with the original car's trajectory. The data we use is from the Next Generation Simulation program [45], and is accessible on the U.S. Department of Transportation's website. In Fig. 11, we present two examples where the level-k driver makes the same action decision as the original human driver, although there is a slight mismatch in the exact motion, which is most likely due to a difference between our EOM and the actual dynamics of that specific vehicle. In the left column, the level-1 test car (the red car) and the original car in the data (indicated by the blue box) both make a lane change from the right lane to the left lane, because the front car in the right lane is approaching fast. In the right column, the level-2 test car and the original car in the data both make a lane change from the middle lane to the right lane, because the front car in the

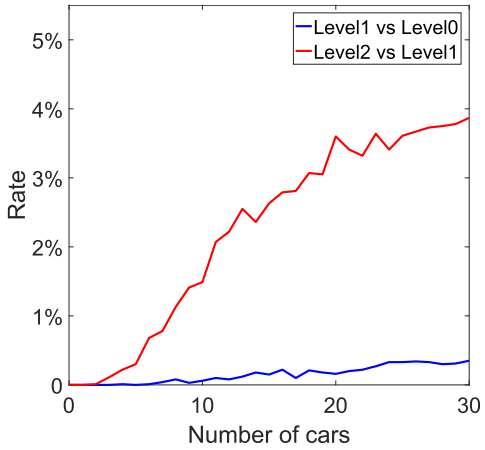


Fig. 12. Constraint violation rates of the level-1 and level-2 policies.

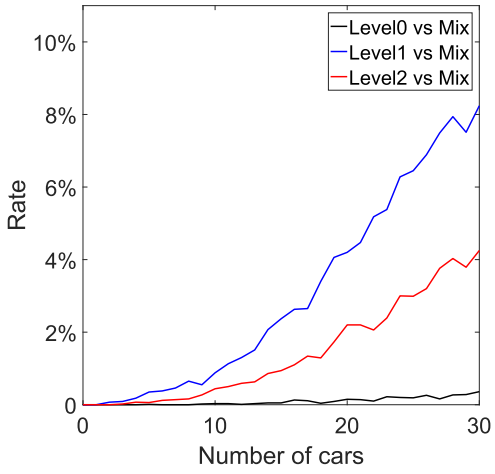


Fig. 13. Constraint violation rates of the level-0, level-1, and level-2 policies in the same mixed traffic.

middle lane is approaching, while the front car in the right lane is moving away.

It is worth mentioning that it is in general very difficult to come to conclusions about the validity of driver models [33]. However, reasonable maneuver behaviors are observed in our simulations of the level-k driver models. More comprehensive validation of the model is left to the future work.

Fig. 12 shows the constraint violation rates of the level-1 and level-2 test cars for varying numbers of cars in the traffic. Here, "constraint violation" refers to a violation of the safe zone of the test car by any of the vehicles in the simulations. To obtain these rates, 10 000 simulations are run for each case (i.e., for each value of the number of cars). Each simulation lasts 200 s and the rates are provided as the percentage of simulation runs during which safe zones are violated. Note that the level-2 test car is placed in traffic formed of level-1 vehicles, and the level-1 test car is placed in an environment of level-0 vehicles. It is seen that the level-2 test car experiences higher violation rates than the level-1 test car in this experiment. One explanation for this is that the traffic flow consisting of level-1 cars, where the level-2 policy is evaluated, is much harder to predict compared with the one consisting of level-0 cars, where the level-1 policy is tested.

Fig. 13 shows the constraint violation rates of the level-0, level-1, and level-2 test cars in the same traffic environment.

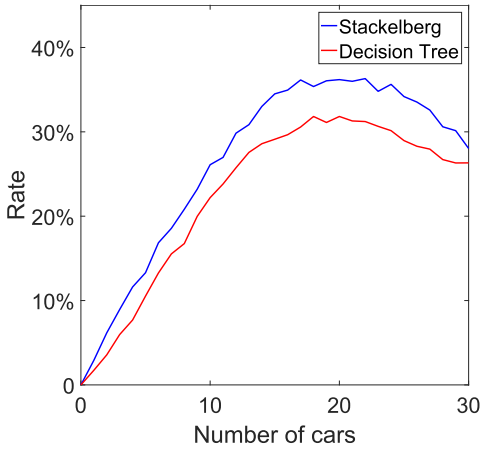


Fig. 14. Constraint violation rates of the Stackelberg and decision tree policies.

The traffic is modeled by mixing 10% level-0 drivers, 60% level-1 drivers, and 30% level-2 drivers (the same traffic environment used for testing the Stackelberg and decision tree policies in the next section). We observe that the level-0 policy has the lowest constraint violation rates. This is due to the fact that the level-0 policy in Section III-A is based on a very conservative control logic, e.g., never making lane changes or accelerations to overtake other cars. Comparing the level-1 policy and the level-2 policy, we can observe that the level-2 policy has better safety performance in this mixed traffic. One explanation for this is that the level-2 policy is trained in a more complicated traffic flow of level-1 cars compared with the traffic flow of level-0 cars used to train the level-1 policy. As a result, a level-2 driver may be able to handle unfamiliar traffic flows better than a level-1 driver.

E. Comparative Quantitative Evaluation of Stackelberg and Decision Tree Policies

At first, we test the Stackelberg and the decision tree policies in a traffic consisting of only level-0 vehicles. A defining feature of the level-0 policy is that the vehicles do not change lanes. It is observed that both the Stackelberg and the decision tree policies perform well in this environment, i.e., constraint violations are not observed. This is also in agreement with the results in [10], [15], and [16]. The figures are omitted as they provide no additional information.

Next, we consider a simulated traffic environment where 10% of the drivers make decisions based on level-0 policies, 60% of the drivers act based on level-1 policies, and 30% use level-2 policies. These percentages of various levels are assumed based on an experimental study conducted in [37]. Fig. 14 shows the distance constraint violation rates for the Stackelberg and decision tree policies versus the number of cars in the traffic. Each simulation is 200 s long, and 10 000 simulations are run for each case (i.e., for each value of the number of cars).

As seen in Fig. 14, both approaches exhibit significant distance constraint violation percentages. Note that these violations could also be caused by our level-k drivers, but as shown in Figs. 12 and 13, compared with the numbers in Fig. 14, the constraint violation rates of the level-k policies are small.

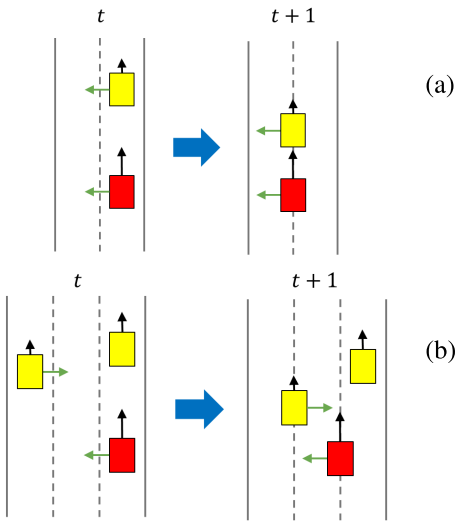


Fig. 15. Scenarios for distance constraint violation. (a) A constraint violation scenario on a two-lane highway. (b) A constraint violation scenario on a three-lane highway.

The main reason for the distance constraint violations is that the developed traffic model with interacting drivers is more complex than the traffic models used for the development of the Stackelberg and decision tree algorithms. To the best of our knowledge, few works addressing the development of autonomous vehicle control policies consider traffic of similar complexity.

Fig. 15 shows two cases of driver interactions that are responsible for many distance constraint violations. The red rectangle indicates the car being tested, while the yellow rectangles are the cars in the traffic environment. The black arrows indicate the longitudinal velocities of the cars, while the green arrows indicate the lateral velocities. The left-hand side presents the scenarios at time t, while the right-hand side presents the scenarios at time $t+1$. In Fig. 15(a), the red car is starting to change lane to the left, trying to overtake the front car, while at the same time the front car is also starting to change the lane because of some other cars ahead of it (not shown in this figure). As a result, both cars are changing lanes while their longitudinal distance keeps decreasing until the safe distance constraint is violated. Although the red car may brake hard after the lane change, trying to avoid this distance constraint violation, at that time its range is already too small. In Fig. 15(b), the red car is starting to change lane to the left to overtake the car in front of it. Although the car in front remains in its own lane, there is another car in the leftmost lane starting to move into the middle lane as well. As a result, both the red car and the yellow car in the leftmost lane are changing lanes to the middle and eventually violate the safe distance constraint.

We note that challenging scenarios such as the ones above can greatly facilitate the testing of autonomous driving control algorithms. In fact, the above two cases are also dangerous situations for a human driver. Many traffic accidents result from the driver misjudging the potential actions of the surrounding vehicles. According to the National Highway Traffic Safety Administration's 2008 National Motor Vehicle Crash Causation Survey, human error is the critical reason for 93% of motor vehicle crashes [46].

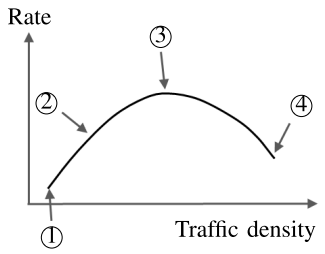


Fig. 16. Diagram of constraint violation rate.

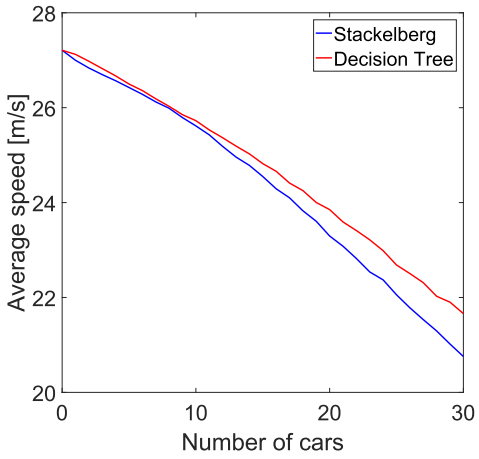


Fig. 17. Average travel speed of the Stackelberg and decision tree policies.

Note also that the constraint violation rate first increases, peaks, and then decreases as the number of cars in the environment, which reflects the traffic density, increases. As Fig. 16 shows, when the traffic is very sparse, cars on the road can drive almost freely and have low chance of having a constraint violation (①). As the number of cars in the traffic increases, the chance of experiencing an incident also increases (②) (until a peak ③). When the traffic becomes very dense, for instance, in a traffic jam, the average travel speed becomes low, and at the same time each car mostly stays in its own lane and has few lane change maneuvers. As a result, the probability of having constraint violations becomes low again (④). This shape also matches the statistical results of the relationship between car crash rates and traffic densities based on real traffic data in [47].

Apart from using the constraint violation rate as a metric to measure the safety and robustness of the Stackelberg and decision tree policies, we also use the average travel speed to measure the driving performance. It can be observed from Fig. 17 that the decision tree policy has better driving performance compared with the Stackelberg policy.

Fig. 18 compares the computational load of the two autonomous driving control methods. The numbers shown in the plot are the average time consumed to run the 200-s-long simulation. Because the decision tree policy needs to evaluate in total 49 two-layer action profiles, its required computational effort is higher than that of the Stackelberg policy. As a summary, our implementation of the decision tree algorithm exhibits better safety and performance characteristics than our

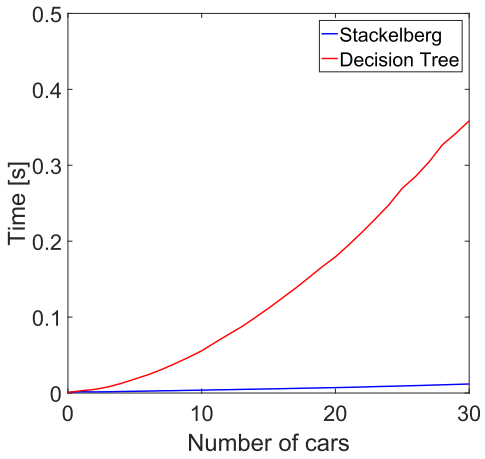


Fig. 18. Computational time of the Stackelberg and decision tree policies.

implementation of the Stackelberg algorithm, while the price paid is the higher computational cost. The numbers in Fig. 18 are obtained using the Java $System.nanoTime()$ function, and the simulations are run on a desktop with i7-4790 3.60GHz CPU and with Eclipse Java Neon platform. Note also that as Fig. 18 indicates, the simulator is quite fast: a 200-s-long simulation of up to 30 interacting agents can be done within tens to hundreds of milliseconds.

F. Optimal Autonomous Driving Controller Calibration

One of the potential uses of the proposed traffic modeling approach is for calibrating parameter values in the autonomous driving control algorithms. We illustrate this using the decision tree policy as an example.

The optimization objective is defined by considering both safety and performance: the goal is to maximize the following reward function:

$$R_{\text{obj}} = p_1(-\bar{c}) + p_2 \frac{\bar{v}_x - v_{\min}}{v_{\max} - v_{\min}} \tag{16}$$

where the weights p_1 and p_2 are determined by the user, $\bar{c}$ is the constraint violation rate defined as in Fig. 14, $\bar{v}_x$ is the average speed during the 200-s-simulations, and $v_{\min}$ and $v_{\max}$ are the lower and the upper bounds of the test vehicle's speed, respectively. Note that this reward function is designed such that each of its terms is dimensionless. The parameters optimized in this example are the ratio $(w_{l1})/(w_{l2})$, representing the weighting of the two layers in the evaluation metric function (13) in the decision tree evaluations, and x_B, the size of region B in the longitudinal direction—a threshold of triggering the path planner. These two parameters are selected for optimization since they have indirect influence on safety and performance, making them difficult to set from intuition. Note that the influence of other parameters, for example, $w_1, \ldots, w_4$, in the decision tree evaluation metric function, is more intuitive and they can be tuned more easily.

Fig. 19 shows the surface of the reward values as a function of $(w_{l1})/(w_{l2})$ and x_B corresponding to different weight selections p_1 and p_2, in the presence of a 20-car traffic. These figures can be used to pick the best pair of $(w_{l1})/(w_{l2})$ and x_B for a given reward function. For example, for maximum

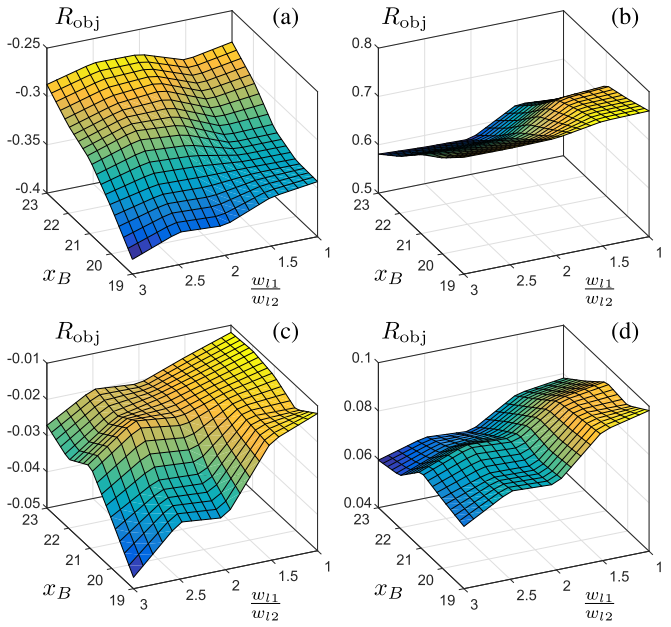


Fig. 19. Parameter optimization results corresponding to different reward function designs. (a) p_1=1, $p_2 = 0$. (b) $p_1 = 0$, $p_2 = 1$. (c) $p_1 = 0.7$, $p_2 = 0.3$. (d) $p_1 = 0.6$, $p_2 = 0.4$.

safety ($p_1 = 1$, $p_2 = 0$), Fig. 19(a) shows that the best pair is $(2.5, 23)$. The rate of constraint violation with this pair is 27.5%; while for the original selection $(2, 21)$ in the previous section, the corresponding violation rate is 31.8%.

VI. Conclusion

In this paper, a hierarchical reasoning game theory-based approach to model interacting driver behavior in traffic was presented. The proposed method provides an approach to simulate interactive driver behavior under the given traffic conditions.

A traffic simulator was developed using level-k driver models. It can be used for testing and verification of autonomous driving algorithms, and for discovery of challenging trajectories and scenarios that can facilitate the testing of future autonomous vehicles. To illustrate the simulator use, we have defined and tested two autonomous vehicle control policies in terms of safety and performance. Our traffic simulator can also be used for parameter calibration of these policies by a simulation-based optimization approach.

References

[1] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray, "Autonomous driving in urban environments: Approaches, lessons and challenges," *Philos. Trans. Roy. Soc. London A, Math. Phys. Sci.*, vol. 368, no. 1928, pp. 4649–4672, 2010.

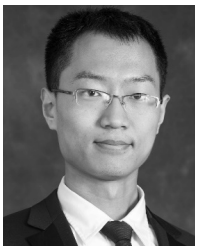
[2] J. M. Anderson, K. Nidhi, K. D. Stanley, P. Sorensen, C. Samaras, and O. A. Oluwatola, *Autonomous Vehicle Technology: A Guide for Policymakers*. Santa Monica, CA, USA: Rand Corporation, 2014.

[3] N. Kalra, "With driverless cars, how safe is safe enough?" The RAND Center Decision Making Under Uncertain., Santa Monica, CA, USA, Tech. Rep., 2016. [Online]. Available: http://www.rand.org/blog/2016/02/with-driverless-cars-how-safe-is-safe-enough.html

[4] T. Wongpiromsarn and R. M. Murray, "Formal verification of an autonomous vehicle system," in *Proc. Conf. Decision Control*, 2008.

[5] T. Wongpiromsarn, S. Mitra, R. M. Murray, and A. Lamperski, "Periodically controlled hybrid systems: Verifying a controller for an autonomous vehicle," in *Proc. HSCC*, Pasadena, CA, USA, 2008.

[6] J. Lygeros, D. N. Godbole, and S. Sastry, "Verified hybrid controllers for automated vehicles," *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 522–539, Apr. 1998.
[7] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 903–918, Aug. 2014.
[8] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, "Automated driving: The role of forecasts and uncertainty—A control perspective," *Eur. J. Control*, vol. 24, pp. 14–32, Jul. 2015.
[9] I. Miller *et al.*, "Team Cornell's Skynet: Robust perception and planning in an urban environment," *J. Field Robot.*, vol. 25, no. 8, pp. 493–527, 2008.
[10] L. Claussman, A. Carvalho, and G. Schildbach, "A path planner for autonomous driving on highways using a human mimicry approach with binary decision diagrams," in *Proc. Eur. Control Conf.*, Linz, Austria, Jul. 2015, pp. 2976–2982.
[11] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs," in *Proc. IEEE 17th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 392–399.
[12] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," in *Robot., Sci. Syst.*, vol. 41, no. 6, pp. 1367–1382, 2015.
[13] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 3, pp. 566–580, May 2007.
[14] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2013, pp. 2335–2340.
[15] J. H. Yoo and R. Langari, "Stackelberg game based model of highway driving," in *Proc. ASME 5th Annu. Dyn. Syst. Control Conf. Joint JSME 11th Motion Vibrat. Conf.*, 2012, pp. 499–508.
[16] J. H. Yoo and R. Langari, "A Stackelberg game theoretic driver model for merging," in *Proc. ASME Dyn. Syst. Control Conf.*, 2013, p. V002T30A003.
[17] S. Lefèvre, Y. Gao, D. Vasquez, H. E. Tseng, R. Bajcsy, and F. Borrelli, "Lane keeping assistance with learning-based driver model and model predictive control," in *Proc. 12th Int. Symp. Adv. Vehicle Control*, 2014.
[18] S. Lefèvre, A. Carvalho, and F. Borrelli, "Autonomous car following: A learning-based approach," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun./Jul. 2015, pp. 920–926.
[19] R. Vasudevan, V. Shia, Y. Gao, R. Cervera-Navarro, R. Bajcsy, and F. Borrelli, "Safe semi-autonomous control with enhanced driver modeling," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2012, pp. 2896–2903.
[20] V. A. Shia *et al.*, "Semiautonomous vehicular control using driver modeling," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2696–2709, Dec. 2014.
[21] D. Salvucci, E. Boer, and A. Liu, "Toward an integrated model of driver behavior in cognitive architecture," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1779, pp. 9–16, 2001.
[22] P. Hidas, "Modelling lane changing and merging in microscopic traffic simulation," *Transp. Res. C, Emerg. Technol.*, vol. 10, no. 5, pp. 351–371, Oct./Dec. 2002.
[23] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Proc. Robot., Sci. Syst. Conf. (RSS)*, 2016.
[24] R. Lee and D. Wolpert, "Game theoretic modeling of pilot behavior during mid-air encounters," in *Decision Making with Imperfect Decision Makers*. Heidelberg, Germany: Springer, 2012, pp. 75–111.
[25] S. Backhaus *et al.*, "Cyber-physical security: A game theory model of humans interacting over control systems," *IEEE Trans. Smart Grid*, vol. 4, no. 4, pp. 2320–2327, Dec. 2013.
[26] C. Dextreit and I. V. Kolmanovsky, "Game theory controller for hybrid electric vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 2, pp. 652–663, Mar. 2014.
[27] Y. Yildiz, A. Agogino, and G. Brat, "Predicting pilot behavior in medium-scale scenarios using game theory and reinforcement learning," *J. Guid., Control, Dyn.*, vol. 37, no. 4, pp. 1335–1343, 2014.
[28] N. Musavi, D. Onural, K. Gunes, and Y. Yildiz, "Unmanned aircraft systems airspace integration: A game theoretical framework for concept evaluations," *J. Guid., Control, Dyn.*, vol. 40, no. 1, pp. 96–109, 2016.
[29] D. W. Oyler, Y. Yildiz, A. R. Girard, N. I. Li, and I. V. Kolmanovsky, "A game theoretical model of traffic with multiple interacting drivers for use in autonomous vehicle development," in *Proc. IEEE Amer. Control Conf. (ACC)*, Jul. 2016, pp. 1705–1710.
[30] N. Li, D. Oyler, M. Zhang, Y. Yildiz, A. Girard, and I. Kolmanovsky, "Hierarchical reasoning game theory based approach for evaluation and testing of autonomous vehicle control systems," in *Proc. IEEE 55th Conf. Decision Control (CDC)*, Dec. 2016, pp. 727–733.
[31] S. Kikuchi and P. Chakroborty, "Car-following model based on fuzzy inference system," *Transp. Res. Rec.*, no. 1365, p. 82, 1992.
[32] M. McDonald, J. Wu, and M. Brackstone, "Development of a fuzzy logic based microscopic motorway simulation model," in *Proc. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Nov. 1997, pp. 82–87.
[33] M. Brackstone and M. McDonald, "Car-following: A historical review," *Transp. Res. F, Traffic Psychol. Behavior*, vol. 2, no. 4, pp. 181–196, 1999.
[34] R. M. Michaels, "Perceptual factors in car following," in *Proc. 2nd Int. Symp. Theory Road Traffic Flow*, London, U.K., 1963, pp. 44–59.
[35] D. Hoefs, "Entwicklung einer messmethode uber den bewegungsablauf des kolonnenverrkehrs," Karlsruhe Inst. Technol., Karlsruhe, Germany, Tech. Rep., 1972.
[36] D. O. Stahl and P. W. Wilson, "On players' models of other players: Theory and experimental evidence," *Games Econ. Behavior*, vol. 10, no. 1, pp. 218–254, Jul. 1995.
[37] M. A. Costa-Gomes, V. P. Crawford, and N. Iriberri, "Comparing models of strategic thinking in van Huyck, Battalio, and Beil's coordination games," *J. Eur. Econ. Assoc.*, vol. 7, nos. 2–3, pp. 365–376, Apr./May 2009.
[38] T. Hedden and J. Zhang, "What do you think I think you think?: Strategic reasoning in matrix games," *Cognition*, vol. 85, no. 1, pp. 1–36, Aug. 2002.
[39] M. A. Costa-Gomes and V. P. Crawford, "Cognition and behavior in two-person guessing games: An experimental study," *Amer. Econ. Rev.*, vol. 96, no. 5, pp. 1737–1768, 2006.
[40] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1. Cambridge, U.K.: MIT Press, 1998, ch. 1.
[41] T. Jaakkola, S. P. Singh, and M. I. Jordan, "Reinforcement learning algorithm for partially observable Markov decision problems," in *Proc. Adv. Neural Inf. Process. Syst.*, 1995, pp. 345–352.
[42] S. Mahadevan, "Average reward reinforcement learning: Foundations, algorithms, and empirical results," *Mach. Learn.*, vol. 22, nos. 1–3, pp. 159–195, 1996.
[43] R. A. Freeman and P. Kokotovic, "Inverse optimality in robust stabilization," *SIAM J. Control Optim.*, vol. 34, no. 4, pp. 1365–1391, 1996.
[44] A. Y. Ng *et al.*, "Algorithms for inverse reinforcement learning," in *Proc. ICML*, 2000, pp. 663–670.
[45] Federal Highway Administration, "NGSIM—Next generation simulation," United States Federal Government, Washington, DC, USA, Tech. Rep., accessed on Jun. 20, 2017. [Online]. Available: http://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm
[46] N. H. T. S. Administration, "2008 national motor vehicle crash causation survey," United States Federal Government, Washington, DC, USA, Tech. Rep. DOT HS 811 059, accessed on Jun. 20, 2017. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811059
[47] D. Lord, A. Manar, and A. Vizioli, "Modeling crash-flow-density and crash-flow-V/C ratio relationships for rural and urban freeway segments," *Accident Anal. Prevention*, vol. 37, no. 1, pp. 185–199, 2005.



Nan Li received the B.S. degree in automotive engineering from Tongji University, Shanghai, China, in 2014, and the M.S. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2016, where he is currently pursuing the Ph.D. degree in aerospace engineering.

His current research interests include model-based and model-free optimization and optimal control, and applications of game theory, and of reinforcement learning to automotive control problems.

Dave W. Oyler received the B.S. degree in electrical engineering from Texas A&M University, College Station, TX, USA, in 2012, and the M.S. and Ph.D. degrees in aerospace engineering from the University of Michigan, Ann Arbor, MI, USA, in 2014 and 2016, respectively.

His current research interests include nonlinear control, differential games, and heterogeneous multiagent systems.

Mengxuan Zhang received the B.S. degree in aerospace engineering from the University of Michigan, Ann Arbor, MI, USA, in 2017, and the B.S. degree in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2017. She is currently pursuing the M.S. degree in aeronautics and astronautics with Stanford University, Stanford, CA, USA.

Yildiray Yildiz (SM'17) received the B.S. degree (valedictorian) in mechanical engineering from Middle East Technical University, Ankara, Turkey, in 2002, the M.S. degree in mechatronics from Sabanci University, Istanbul, Turkey, in 2004, and the Ph.D. degree in mechanical engineering with a minor in mathematics from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2009.

He held post-doctoral associate and associate scientist positions with the NASA Ames Research Center, Mountain View, CA, USA, employed by the University of California, Santa Cruz, CA, USA, from 2009 to 2010 and 2010 to 2014, respectively. He is currently an Assistant Professor with Bilkent University, Ankara, Turkey. His current research interests include adaptive control, time-delay systems, and applications of game theory and machine learning methods to automotive and aerospace modeling and control problems.

Dr. Yildiz was a recipient of the NASA Honor Award, the Turkish Science Academy Young Scientist Award, and the ASME Best Student Conference Paper Award. He is serving as an Associate Editor of the *IEEE Control Systems Magazine*.

Ilya Kolmanovsky (F'08) received the M.S. and Ph.D. degrees in aerospace engineering and the M.A. degree in mathematics from the University of Michigan, Ann Arbor, MI, USA, in 1993, 1995, and 1995, respectively.

He is currently a Full Professor with the Department of Aerospace Engineering, University of Michigan. He is named as an inventor on 97 U.S. patents. His current research interests include control theory for systems with state and control constraints and control applications to aerospace and automotive systems.

Dr. Kolmanovsky was a recipient of the Donald P. Eckman Award of American Automatic Control Council and of two IEEE Transactions on Control Systems Technology outstanding paper awards.

Anouck R. Girard (SM'12) received the Ph.D. degree in ocean engineering from the University of California, Berkeley, CA, USA, in 2002.

She has been with the University of Michigan, Ann Arbor, MI, USA, since 2006, where she is currently an Associate Professor of Aerospace Engineering. She has co-authored the book *Fundamentals of Aerospace Navigation and Guidance* (Cambridge University Press, 2014). Her current research interests include flight dynamics and control systems.

Dr. Girard was a recipient of the Silver Shaft Teaching Award from the University of Michigan and a Best Student Paper Award from the American Society of Mechanical Engineers.