LUCIDGames: Online Unscented Inverse Dynamic Games for Adaptive Trajectory Prediction and Planning

Simon Le Cleac'h, Mac Schwager, and Zachary Manchester

Abstract—**Existing game-theoretic planning methods assume that the robot knows the objective functions of the other agents *a priori* while, in practical scenarios, this is rarely the case. This letter introduces LUCIDGames, an inverse optimal control algorithm that is able to estimate the other agents' objective functions in real time, and incorporate those estimates online into a receding-horizon game-theoretic planner. LUCIDGames solves the inverse optimal control problem by recasting it in a recursive parameter-estimation framework. LUCIDGames uses an unscented Kalman filter (UKF) to iteratively update a Bayesian estimate of the other agents' cost function parameters, improving that estimate online as more data is gathered from the other agents' observed trajectories. The planner then takes account of the uncertainty in the Bayesian parameter estimates of other agents by planning a trajectory for the robot subject to uncertainty ellipse constraints. The algorithm assumes no explicit communication or coordination between the robot and the other agents in the environment. An MPC implementation of LUCIDGames demonstrates real-time performance on complex autonomous driving scenarios with an update frequency of 40 Hz. Empirical results demonstrate that LUCIDGames improves the robot's performance over existing game-theoretic and traditional MPC planning approaches. Our implementation of LUCIDGames is available at https://github.com/RoboticExplorationLab/LUCIDGames.jl.**

Index Terms—**Multi-Robot systems, motion and path planning, intelligent transportation systems.**

I. Introduction

PLANNING trajectories for a robot that interacts with other agents is challenging, as it requires prediction of the reactive behaviors of the other agents, in addition to planning for the robot itself. Classical approaches in the literature decouple the prediction and planning tasks. Usually, predicted trajectories of the other agents are computed first and provided as input for the robot planning module, which considers them as immutable obstacles. This formulation ignores the influence of the robot's decisions on the other agents' behaviors. Moreover, it can lead to the "frozen robot" problem, where no safe path to the goal can be found by the planner [1] because of the false assumption that other agents will not yield or deviate from their predicted trajectory in response to the robot. Preserving the coupling between prediction and planning is thus key to producing richer interactive behavior for a robot acting among other agents.

Several recent works have used the theory of dynamic games to capture the coupled interaction among a robot and other agents, particularly in the context of autonomous driving [2], [3]. However, these works rely on the strong assumption that the robot has full knowledge of the other agents' objective functions. In many applications, the robot only has access to a coarse estimate of these objective functions. For instance, in a ramp-merging scenario, the autonomous car might be aware of the desired distance drivers usually keep between themselves. These coarse estimates of the other agents' objectives can be obtained using real data like the NGSIM driving dataset [4]. Inverse reinforcement learning (IRL) approaches typically learn a general

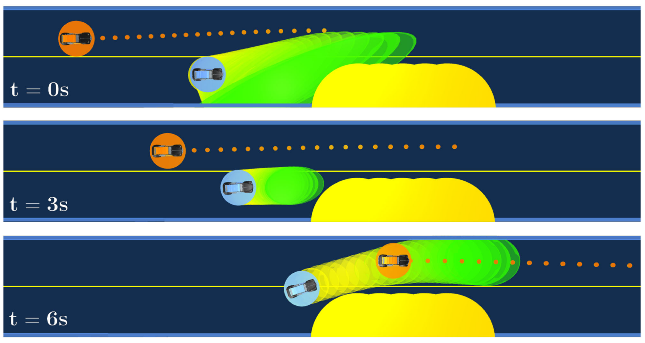

Fig. 1. We present three roadway visualizations of a scenario where the autonomous vehicle (AV) in orange and a human-driven car (blue) have to overtake a large obstacle (yellow) on the bottom lane. The AV (orange) follows the robust version of LUCIDGames. At the start, the AV slows down to avoid the uncertainty-based collision avoidance zone (green), which comprises two possibilities: either the human cuts in front of the AV, or the human yields to let the AV go first. Then, by observing the human's behavior, the AV better estimates its objective and narrows down the collision avoidance zone to the first option. Finally, the AV proceeds to overtaking the obstacle before the human. The AV's planned trajectory is represented by orange dots.

Manuscript received October 14, 2020; accepted February 20, 2021. Date of publication April 21, 2021; date of current version May 19, 2021. This letter was recommended for publication by Associate Editor H. Kurniawati and Editor N. Amato upon evaluation of the reviewers' comments. This work was supported in part by NSF NRI under Grant 1830402, in part by DARPA YFA under Grant D18AP00064, and in part by ONR under Grant N00014-18-1-2830. Toyota Research Institute ("TRI") provided funds to assist the authors with their research, but this article solely reflects the opinions, and conclusions of its authors, and not TRI or any other Toyota entity. *(Corresponding author: Simon Le Cleac'h.)*

Simon Le Cleac'h is with the Department of Mechanical Engineering, Stanford University, California, CA 94305 USA (e-mail: simonlc@stanford.edu).

Mac Schwager is with the Department of Aeronautics & Astronautics, Stanford University, California, CA 94305 USA (e-mail: schwager@stanford.edu).

Zachary Manchester is with the Robotics Institute, Carnegie Mellon University, Pennsylvania, PA 15213 USA (e-mail: zacm@cmu.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3074880, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3074880

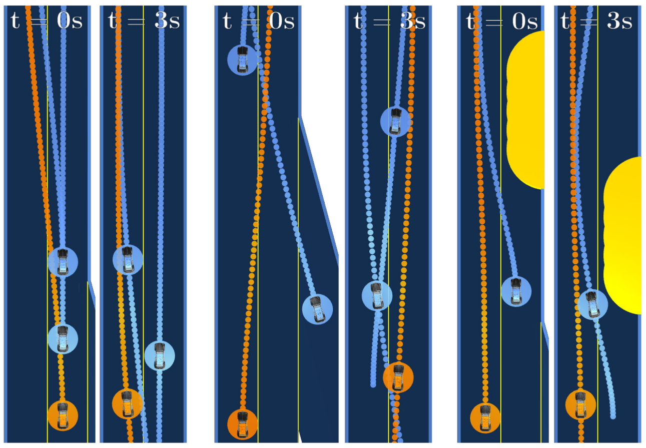

Fig. 2. From left to right, we represent an overtaking maneuver, a ramp merging, and an obstacle avoidance maneuver, the robot is shown in orange.

objective function to suit a large batch of demonstrations from multiple agents [5]–[7]. On the contrary, our goal is to accurately estimate the individualized objective functions of specific agents in the vicinity of the robot we control. This online estimation process occurs while interacting with the other agents so that the robot can adapt to each agent individually. For instance, our approach allows for estimating the level of aggressiveness of a specific driver in the surroundings of the autonomous car.

The online estimation approach we propose is complementary to classical offline IRL methods: With IRL, we can learn a relevant set of objective-function features from real data, as well as a prior distribution over the objective-function parameters. Given these features and a prior on the parameters, we can use LUCIDGames to refine the parameter estimation for a specific agent based on online observation of this agent. Our approach assumes that agents solve a dynamic game and follow Nash equilibrium strategies. This setting models non-cooperating agents that act optimally to satisfy their individual objectives [2], [3]. We further assume that we have access to a class of objective functions parameterized by a small number of parameters. This could be the desired speed, or driver aggressiveness in the autonomous driving context.

To estimate these parameters, we adopt the unscented Kalman filtering (UKF) approach. In our case, the key part of this algorithm is the measurement model that maps the objective function parameters to the observation of the surrounding agents' next state. To obtain this mapping, we use ALGAMES, a trajectory optimization solver for dynamic games that handles general nonlinear state and input constraints [2].The choice of a derivative-free estimation method (UKF) is justified by the complexity of the measurement model, which includes multiple non-convex constrained optimization problems. Additionally, we design a planner for the robot that is robust to poor estimates of the other agents' objectives. By sampling from the belief over the objective functions of the other agents and computing trajectories corresponding to those samples, we can translate the uncertainty in objective functions into uncertainty in predicted trajectories. Then, ellipsoidal bounds are fitted to the sampled trajectories to form "safety constraints"; collision constraints that account for objective uncertainty. Importantly, the calculation of these safety constraints reuses samples required by the UKF estimation algorithm. It is, therefore, executed at a negligible additional cost.

# of Players	Freq. in Hz	$\mathbb{E}[\delta t]$ in ms	$\sigma[\delta t]$ in ms
2	132	7.55	16.7
3	38.0	26.3	37.7
4	11.4	87.3	94.0

Fig. 3. Running the MPC implementation of LUCIDGames 50 times on a ramp-merging scenario with 2, 3 and 4 players, we obtain the mean update frequency of the MPC as well as the mean and standard deviation of δt, the time required to update the MPC plan and the belief over the other players' objective functions.

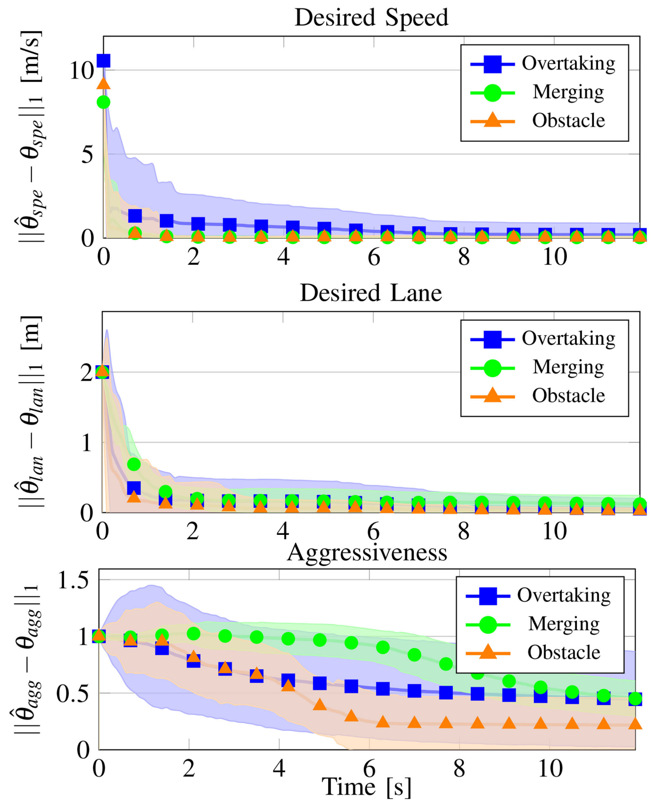

Fig. 4. LUCIDGames reduces the estimation error on the desired speed parameter by a factor of 100 within 12 seconds of interaction (top). The error on the desired lane is divided by 20 (middle) and the error on the aggressiveness parameter is halved (bottom). The markers indicate the median error computed over 50 simulations. The faded color areas correspond to the 95% confidence interval.

In a receding-horizon loop, LUCIDGames controls one agent called the "robot" and estimates the other agents' objectives at 40 Hz for a 3-player game with a strong level of interaction among the agents. Our primary contributions are as follows:

1) We propose a UKF-based method for a robot to estimate the objective function parameters of non-cooperating agents online, and show convergence of the estimate to the ground-truth parameters.(Fig. 4).
2) We combine the online parameter estimator with a game-theoretic planner. The combined estimator and planner, called LUCIDGames, runs online at 40 Hz in a receding-horizon fashion.
3) We include safety constraints within LUCIDGames to impose ellipsoidal collision-avoidance constraints for the robot that reflect the uncertainty in the other agents' future trajectories due to the Bayesian estimate of their parameters.

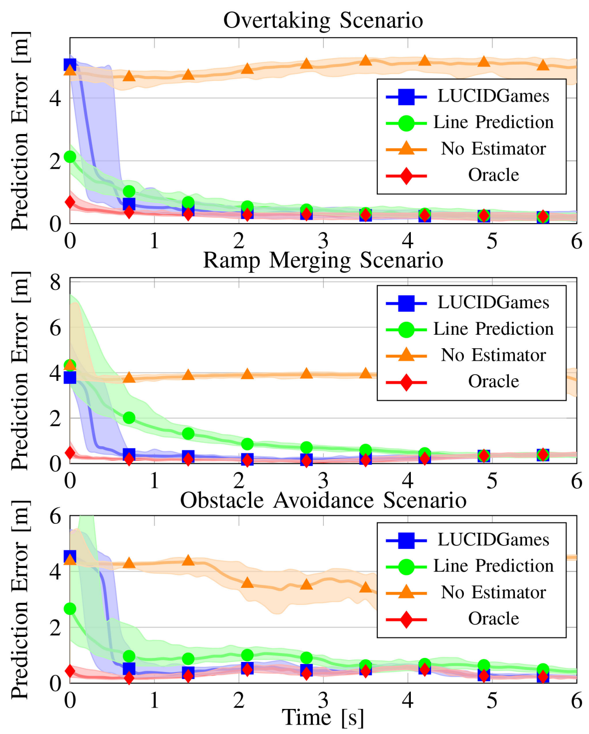

Fig. 5. We present the trajectory prediction error obtained by the robot on 3 scenarios for 4 algorithms: LUCIDGames; a non-game-theoretic baseline using straight-line predictions; a game-theoretic solver that does not estimate the other agents' objective functions and finally, an oracle having access to the ground-truth objective functions of the other agents. LUCIDGames starts with a large prediction error and quickly estimates the other agents' objective functions to outperform the baselines and reach error levels comparable to those of the oracle. We represent the medians, computed over 50 simulations, of the prediction error measuring the $\ell 2$-distance between the 3-second trajectory predictions and the ground-truth trajectory. The faded color areas correspond to the interval between the 1st and 3rd quantile.

We compare LUCIDGames against game-theoretic and non-game-theoretic baselines. We show that LUCIDGames' trajectory-prediction error rapidly decreases to match the accuracy of the oracle predictor that has access to the ground-truth objectives (Fig. 5). Furthermore,

LUCIDGames with safety constraints allows for realistic, cautious interactions between a robot and an agent making an unexpected maneuver (Fig. 1).

II. Related Work

A. Game-Theoretic Trajectory Optimization

Dynamic games have been used as a modeling framework in a wide variety of applications. For example, in autonomous driving [2], [3], drone and car racing [8] etc. The solutions of a dynamic game depend on the type of equilibrium selected [9]. Nash equilibrium models games without hierarchy between players. Each player's strategy is the best response to the other players' strategies.

Nash equilibrium solutions have been studied extensively [3], [8], [10]. They seem to capture the game-theoretic interactions observed in some multi-agent non-cooperative problems. We follow this approach by solving for open-loop Nash equilibrium strategies. However, we intend to relax a key assumption made in previous works by estimating the other agents' objective functions instead of assuming that they are known *a priori* by the robot we control.

B. Objective Function Estimation

Estimating objective functions from historical data is a well investigated problem known as Inverse Optimal Control (IOC) or Inverse Reinforcement Learning (IRL). Typically, with the IRL approach, the objective function is linear in terms of a given set of state features [6]. The goal is to identify a parameter vector that weights these features so that the behavior resulting from this estimated objective matches the observed behavior. While these classical approaches are usually framed in the discrete state and action space setting, they can also be applied to continuous state and action spaces arising in robotics problems [11], [12]. However, these works are limited to single-agent problems.

In the multi-agent setting, some IRL approaches formulate the problem by assuming cooperative agents [13] or competing agents [7], [14]. These approaches have been demonstrated on discretized state and actions spaces. More recent works consider the multi-agent competitive setting with continuous state and action spaces [15], [16]. Their methods have typically been demonstrated on linear-quadratic games with low-dimensional states and control inputs. IOC and IRL-based techniques estimate the objective function's parameters "offline". Given a set of complete trajectories, they intend to identify one parameter vector that will best fit the data. Our goal is slightly different: As an agent in the game, we would like to perform the estimation "online," with only knowledge of previous steps, and use our estimate to inform our actions for future time steps. This means that we have access to fewer demonstrations and that our computation time is limited to ensure real-time execution. On the other hand, we assume a low-dimensional parameter space with a coarse prior.

C. Online Parameter Estimation

Our goal is to estimate the objective functions of the agents in the robot's surroundings. We assume that these objectives are parameterized by a vector of *quasistatic* parameters, denoted θ. This means that θ fluctuates at a rate significantly slower than the filter's update rate (40 Hz). Indeed, these parameters encode drivers' objectives such as desired speed or aggressiveness level, that vary over periods on the order of tens of seconds to minutes. In such situation, a common technique is to perform *parameter estimation* [17] that relies on a stationary process model; as opposed to *state estimation*. In the autonomous driving context, parameter estimation is commonly performed using this stationary process model [18]–[23]. The quasistatic nature of the parameters has been verified empirically on a large driving dataset [18]. This design choice is additionally motivated by the desire to make minimal assumptions the about how the drivers' objectives will evolve over time [23]. Previous works demonstrated that estimating the other drivers' objectives helps better predict their future trajectories. However, this gained information was not used to improve the decision making of the cars. Indeed, Sunberg *et al.* [21], [22] showed that an agent inferring the objectives of the surrounding drivers improves its objective function satisfaction. We also follow a parameter estimation approach, using an UKF with a stationary process model and a measurement model that relies on a game-theoretic modeling of vehicle interactions instead of a rule-based driver model. Additionally, LUCIDGames exploits the gained information to improve the actions of the robot.

LUCIDGames relies on a UKF parameter estimator that propagates in time the belief over the unknown objective parameter vector θ. The procedure that updates this belief is described in Algorithm 1. Line 2 corresponds to the prediction step, which exploits the stationary process model corresponding to an additive white Gaussian noise defined as follows,

$$\theta_{t+1} = \theta_t + \delta_t, \qquad \delta_t \sim \mathcal{N}(0, Q_t). \tag{1}$$

Line 3 samples sigma-points from a Gaussian distribution over the vector θ (2–8),

$$\lambda = \alpha^2(q+\kappa) - q, \tag{2}$$

$$\Theta^{(0)} = \mu, \tag{3}$$

$$\Theta^{(i)} = \mu + (\sqrt{(q+\lambda)\Sigma})_i, \qquad \forall i \in \{1,\ldots,q\} \tag{4}$$

$$\Theta^{(i)} = \mu - (\sqrt{(q+\lambda)\Sigma})_{i-q}, \qquad \forall i \in \{q+1,\ldots,2q\} \tag{5}$$

$$M^{(0)} = \lambda/(q+\lambda), \tag{6}$$

$$C^{(0)} = \lambda/(q+\lambda) + (1-\alpha^2+\beta), \tag{7}$$

$$M^{(i)} = C^{(i)} = \lambda/\left(2(q+\lambda)\right). \qquad \forall i \in \{1,\ldots,2q\} \tag{8}$$

We follow a classical sampling scheme that relies on parameters, α, β, κ, controlling the spread of the sigma-points [24]. Line 4 applies the measurement model that is specific to LUCIDGames to the sampled sigma-points, Θ. R denotes the covariance of the Gaussian white noise associated with the measurement model. Lines 5 to 8 compute the Kalman gain, K, and measurement prediction $\bar{x}_t$. Finally, the update step is executed in line 9.

Algorithm 1: Parameter Estimation Module.

```
 1: procedure Estimator($x_{t-1}$, $x_t$, $\mu_{t-1}$, $\mu_t$, $\Sigma_t$
 2:     $(\bar{\mu}_{t+1}, \quad \bar{\Sigma}_{t+1}) \leftarrow (\mu_t, \quad \Sigma_t + Q_t))$
 3:     $\Theta, M, C \leftarrow$ SigmaPoints($\bar{\mu}_{t+1}, \bar{\Sigma}_{t+1}$) $\triangleright$ Eq. 2–8
 4:     $\chi^{(i)} \leftarrow$ MeasurementModel($x_{t-1}, \mu_{t-1}, \Theta^{(i)}$) $\forall i$
 5:     $\bar{x}_t \leftarrow \sum_i M^{(i)} \chi^{(i)}$
 6:     $P \leftarrow \sum_i C^{(i)} [\chi^{(i)} - \bar{x}_t][\chi^{(i)} - \bar{x}_t]^T + R$
 7:     $S \leftarrow \sum_i C^{(i)} [\Theta^{(i)} - \bar{\mu}_{t+1}][\chi^{(i)} - \bar{x}_t]^T$
 8:     $K \leftarrow SP^{-1}$
 9:     $(\mu_{t+1}, \quad \Sigma_{t+1}) \leftarrow$
        $(\bar{\mu}_{t+1} + K(x_t - \bar{x}_t), \quad \bar{\Sigma}_{t+1} - KPK^T)$
10:     return $\mu_{t+1}, \Sigma_{t+1}$
```

D. Data-Driven Trajectory Prediction

There is a rich literature on applying data-driven approaches to pedestrian or vehicle trajectory predictions. Such approaches are usually trained offline as a general model to suit multiple agents. Additionally, they require a large corpus of data ranging from a thousand driving scenarios [25], [26], to millions of examples [27]. On the other hand, we require a small amount of data and find parameters online for a specific agent. Contrary to deep-learning approaches, our method adapts in real-time to each vehicle by reasoning about its individual objective and by exploiting data gathered online. Offline data-driven methods are complementary to our approach as they could provide a strong prior on the parameters estimated online; a prior grounded on real driving behavior. Data-driven methods can predict a distribution over future trajectories e.g., offline inverse optimal control with online goal inference [28]; Conditional Variational Autoencoders (CVAE) [25] or Generative Adversarial Networks (GAN) [29]. Our approach maintains a unimodal belief over objective function parameters,[1] which translates into a distribution over trajectory predictions. A shortcoming of the CVAE-based or GAN-based methods is that they ignore kinodynamic constraints on the predicted trajectories, allowing cars to move sideways, for instance. Incorporating information like drivable area maps which are common for autonomous driving applications [30], could prevent infeasible trajectory predictions [27]. Our approach generates dynamically feasible and collision-free predictions. One notable work in this field is Trajectron++ [26]. It handles kinodynamic constraints and incorporates drivable area maps, as well as the robot's planned trajectory, to inform the prediction. However, contrary to our algorithm, these data-driven methods ignore collision-avoidance constraints between agents and predict trajectories involving collisions as observed by Bhattacharyya [19] with a learning-based method [29].

[1]Our approach can easily be extended to multimodal belief representation of objective function parameters using a Gaussian mixture model.

III. Problem Statement

A. Guiding Assumptions

In a multi-player dynamic game, the robot takes its control decisions using LUCIDGames and carries out all the computation required by the algorithm. We assume the other agents are "ideal" players in the game. They have access to the ground-truth objective functions of all the players in the game. They take their control decisions by individually solving for a Nash equilibrium strategy based on these true objective functions and execute them in a receding-horizon loop. This assumption is required to avoid the complexity of the robot having to "estimate the estimates" of the other agents. Nevertheless, our algorithm shows strong practical performance even, when this assumption is violated. All the experimental results presented in this letter are obtained with the "ideal" agents having incorrect estimates of the objectives of the surrounding agents in the scene. Moreover, this assumption is way to generate a human driver model that is reactive to the robot's actions and that maintains coupling between planning and trajectory prediction for the robot. Other approaches replayed prerecorded driving data to emulate human driving behavior [18], [19], [23], but this method ignores the reactive nature of human drivers to the robots' decisions. Lane-following models, such as IDM [31] fail to capture complex driving strategies like nudging or changing lanes that our model can generate. We further assume that both the robot and the ideal agents plan by computing open-loop Nash equilibrium trajectories and execute these planned trajectories in a receding horizon loop.

Additionally, to keep the estimation process tractable online, we assume that the agents' objective functions belong to a known class of functions parameterized by a reduced set of quasistatic parameters, as detailed in Section III-C.

B. Dynamic Game Nash Equilibrium

We focus on the discretized dynamic game setting with N time steps and M players (1 robot and $M-1$ agents). We denote $x_k \in \mathbb{R}^n$ the joint state of the system, and $u_k^\nu \in \mathbb{R}^{m^\nu}$ the control input of player ν at time step k. Player ν's strategy is a control input sequence $U^\nu = [(u_1^\nu)^T \dots (u_{N-1}^\nu)^T]^T \in \mathbb{R}^{\bar{m}^\nu}$ where $\bar{m}^\nu = (N-1)m^\nu$. The robot's strategy is denoted, U^r, with $r \in \{1, \dots, M\}$ and U^{-r} designates the *combined* strategies of the $M-1$ other agents in the game. The state trajectory is defined as $X = [(x_1)^T \dots (x_N)^T]^T \in \mathbb{R}^{\bar{n}}$ where $\bar{n} = Nn$. It stems from executing the control strategies of all the players in the game on a joint dynamical system,

$$x_{k+1} = f(x_k, u_k^1, \dots, u_k^M) = f(x_k, u_k), \tag{9}$$

with k denoting the time step index. We define the objective function of player ν; $J^\nu(X, U^\nu) : \mathbb{R}^{\bar{n}+\bar{m}^\nu} \mapsto \mathbb{R}$. It is a function of its strategy, U^ν, and of the state trajectory of the joint system, X. The goal of player ν is to select a strategy, U^ν, that will minimize its cost, J^ν, while respecting kinodynamic and collision-avoidance constraints defined in Sections V.1 and V.2. We compactly express these constraints as a set of inequalities $C : \mathbb{R}^{\bar{n}+\bar{m}} \mapsto \mathbb{R}^{n_c}$:

$$\begin{aligned} \min_{X, U^\nu} \quad & J^\nu(X, U^\nu), \\ \text{s.t.} \quad & C(X, U) \leq 0. \end{aligned} \tag{10}$$

Finding a Nash-equilibrium solution to the set of M Problems (10) is called a generalized Nash equilibrium problem (GNEP) [2], [32]. It consists of finding an open-loop Nash equilibrium control trajectory, i.e. a vector, $\hat{U}$ such that, for all $\nu = 1, \dots, M$, $\hat{U}^\nu$ is a solution to (10) with the other players' strategies set to $\hat{U}^{-\nu}$. This implies that at a Nash equilibrium point, $\hat{U}$, no player can decrease their objective function by unilaterally modifying their strategy, U^ν, to any other feasible point. Solving this GNEP can be done efficiently with a dynamic game solver such as ALGAMES [2]. We will consider it as an algorithmic module that takes as inputs the fully observed initial state of the system and the estimated objective functions, $J^1, \dots, J^M$, and returns an open-loop trajectory of the joint system comprising the robot and the ideal agents.

C. Objective Function Parameterization

As is typically the case in the IRL and IOC literature [6], [12], [15], we assume that the objective function of player ν can be expressed as a linear combination of features, ϕ, extracted from the state and control trajectories of this player,

$$J^\nu(X, U^\nu) = \phi(X, U^\nu)^T \theta^\nu. \tag{11}$$

While restrictive, this parameterization encompasses many common objective functions like linear and quadratic costs. The UKF estimates the weight vector θ^ν of all the agents in the game. We denote by $\theta \in \mathbb{R}^q$ the concatenation of the vectors θ^ν that the robot has to estimate,

$$\theta = [{\theta^1}^T \dots \ {\theta^{r-1}}^T, {\theta^{r+1}}^T \dots \ {\theta^M}^T]^T \in \mathbb{R}^q. \tag{12}$$

IV. Unscented Kalman Filtering Formulation

LUCIDGames allows the robot to estimate the objective functions' parameter θ and to exploit this estimation to predict the other agents' behaviors and make decisions for itself. We represent the belief over the parameter θ as a Gaussian distribution and we sample sigma-points from it. Each sigma-point is a guess over the parameter θ. Given the current state of the system, x, we can form a GNEP for each sigma-point. By solving these GNEPs, we obtain a set of predicted trajectories for the system. When we receive a new measurement of the state x, we compare it to the trajectories we predicted earlier. The Gaussian belief over θ is updated with the typical UKF [24] update rules, so that the sigma-points that had better prediction performance are now more likely.

The UKF framework requires two pieces: the process model, and the measurement model. In a typical filtering context, these are obvious. However, in our problem these are more subtle. Specifically, the quantity we estimate with the filter is θ. Following the parameter estimation framework rationalized in Section II-C, we use a stationary process model where θ evolves according to a random walk (1). The crucial part of our algorithm is the measurement model which is inserted into the classical UKF parameter estimation algorithm. In our case, the measured quantity available to the robot (with noise) is the full system state, x_t, at the current time. Hence, the measurement model is the map relating the parameter vector θ to the system state x_t. This function is itself the solution of the dynamic game. Our UKF, thus, requires the solution of the dynamic game for each sigma-point of θ at each time step.

A. Measurement Model

Our estimator is executed in a receding-horizon loop. At each time step t, the robot updates its Gaussian belief over the vector θ, which is parameterized by its mean, μ_t, and covariance matrix, Σ_t.

We construct a measurement model, $g(\cdot,\cdot)$, that maps the parameter θ and the observed previous state x_{t-1} to the current state x_t that we observe:[2]

$$x_t = g(\theta, x_{t-1}) + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, R). \tag{13}$$

This nonlinear function, $g(\cdot,\cdot)$, encapsulates the decision making process of the agents and the propagation of the system's dynamics as detailed in Algorithm 2. It models the vehicles as game-theoretic agents optimizing their own objective functions. This sets our method apart from online estimation methods relying on rule-based driver models.

B. LUCIDGames: Combining Parameter Estimation and Planning

LUCIDGames exploits the information gained via the estimator to inform the decision making of the robot. It jointly plans for itself and predicts the other agents' trajectories. At time step t, the robot solves the GNEP using the current state of the system x_t and its mean estimate μ_t over θ. We obtain the next state x_{t+1} by propagating forward the open-loop plans of both

[2]A direction for future work could be to consider the case where the robot has a nonlinear or partial observation of the state.

Algorithm 2: Game-theoretic Decision Making.

```
1: procedure MeasurementModel ($x_t$, $\mu_t$, $\theta$)
2:     $U_t^r \leftarrow$ ALGAMES($x_t$, $\mu_t$) $\triangleright$ Robot's plan
3:     $U_t^{-r} \leftarrow$ ALGAMES($x_t$, $\theta$) $\triangleright$ Ideal agents' plans
4:     $U_t \leftarrow [U_t^{rT}, U_t^{-rT}]^T$
5:     $x_{t+1} \leftarrow$ Dynamics($x_t$, $U_t$) $\triangleright$ Equation 9
6:     return $x_{t+1}$
```

Algorithm 3: Combined estimator and planning module.

```
1: procedure LUCIDGames ($x_{t-1}, x_t, \mu_{t-1}, \mu_t, \Sigma_t$)
2:     for $t = 1, 2, \dots$ do
3:     $x_{t+1} \leftarrow$ MeasurementModel($x_t$, $\mu_t$, $\theta$)
4:     $\mu_{t+1}, \Sigma_{t+1} \leftarrow$ Estimator($x_{t-1}, x_t, \mu_{t-1}, \mu_t, \Sigma_t$)
5:     return $x_{t+1}, \mu_{t+1}, \Sigma_{t+1}$
```

the robot and the ideal agents as detailed in Algorithm 2. The joint estimation and control procedure is detailed in Algorithm 3.

V. Simulations: Design and Setup

We apply our algorithm to highway autonomous driving scenarios involving a high level of interactions between agents: overtaking, ramp merging and obstacle avoidance maneuvers (Fig. 2). We assume the robot follows the LUCIDGames algorithm for its decision making and estimation. The other vehicles are modeled as ideal agents solving the dynamic game with knowledge of the true parameters.

1) Problem Constraints: We consider a unicycle model for the dynamics of each vehicle. The state, x_k, contains a 2D position, a heading angle and a scalar velocity for each vehicle. The control input, u_k^ν, consists of an angular velocity and a scalar acceleration. Additionally, we model the collision avoidance zone of each vehicle as a disk, preventing collision between vehicles and with the boundaries of the road.

2) Objective Function: We select a quadratic objective function incentivizing the agents to reach a desired state, x_f, while limiting control inputs. On top of this objective function, we add a quadratic penalty on being close to other vehicles,

$$\begin{aligned} J^\nu(X, U^\nu) = &\sum_{k=1}^{N-1} \frac{1}{2}(x_k - x_f)^T Q (x_k - x_f) + \frac{1}{2} u_k^{\nu T} R u_k^\nu \\ &+ \frac{1}{2}(x_N - x_f)^T Q_f (x_N - x_f) \\ &+ \sum_{k=1}^{N} \sum_{\mu \neq \nu} \gamma^\nu \left(\max\left(0, ||p_k^\nu - p_k^\mu||_2 - \eta(r^\nu + r^\mu)\right)\right)^2. \end{aligned} \tag{14}$$

For agent ν, p_k^ν and r^ν designate its 2D position at time step k and collision avoidance radius. γ^ν and η are scalar collision avoidance cost parameters encoding the magnitude of the cost and the distance at which this cost is "activated".

In this work, we estimate a reduced number of objective function parameters. We choose 3 parameters with intuitive interpretations. Two of them are elements of the desired state, x_f^ν: the desired speed and desired lateral position on the roadway (i.e. desired lane) of the vehicle. The last one is γ^ν, which encodes the "aggressiveness" of the driver. Indeed, a large value for γ^ν will penalize a vehicle driving too close to other vehicles, which will lead to less aggressive behavior. This parameterization is consistent with an objective function expressed as a linear combination of features (11). Therefore, it would be possible to use an IRL algorithm trained on real driving data to provide a prior on these parameters.

VI. Simulation Results

We first assess the tractability and scalability of the approach for an increasing number of agents on highway driving scenarios as shown on Figure 2. Then, we perform an ablation study by removing the two main components of LUCIDGames: the online estimation and the game-theoretic reasoning. The goal is to investigate how each of these components affect the performance of LUCIDGames. This is also a way to compare LUCIDGames to related approaches proposed in the literature. Indeed, several works applied dynamic game solvers in a receding-horizon loop to autonomous driving problems without resorting to online estimation [2], [3]. Bhattacharyya *et al.* [19] used highway driving datasets to compare the trajectory prediction performance of rule-based and black-box driver models. The set of evaluated methods included: constant velocity prediction, Generative Adversarial Imitation Learning (GAIL) [29] and a particle filter estimating the parameters of the intelligent driver model (IDM) online. On the trajectory prediction task, the constant velocity baseline was the best performing method. Thus, we choose to compare our method to this non-game-theoretic baseline.

A. Tractability

We run LUCIDGames in a receding horizon-loop using a coarse prior on the vector θ. In practice, the initial belief is a Gaussian parameterized by its mean and variance:

$$\mu_0 = \mathbf{1}, \qquad \Sigma_0 = v_0 I_q. \tag{15}$$

Where v_0 is a large initial variance on each parameter (typically $v_0 = 25$ in our experiments); and I_q is the identity matrix. We run LUCIDGames on the ramp-merging scenario involving 2 to 4 agents and we compile the timing results in Table 3. We demonstrate the tractability of the algorithm for complex autonomous driving scenarios, and we show real-time performance of the estimator for three agents (40 Hz) and up to four agents (10 Hz). In practice, we trivially parallelize the implementation of LUCIDGames: For each sigma-point, $\Theta_t^{(i)}$, the algorithm requires the solution of a dynamic game (Algorithm 1, line 5). We solve these dynamic games simultaneously, in parallel, by distributing them on a multi-core processor. The number of dynamic games to solve in parallel scales like the number of sigma-points, which is linear in terms of the number of agents M. Each individual dynamic game has a computational complexity of $O(M^3)$. In this work, all the experiments have been executed on a 16-core processor (AMD Ryzen 2950X).

B. Parameter Estimation

We assess the ability of LUCIDGames to correctly estimate the ground-truth objectives of the other agents with only a few seconds of driving interaction on three scenarios: highway overtaking, ramp merging and obstacle avoidance. We compute the relative error between the ground-truth parameter θ and the mean of the Gaussian belief μ_t along the 12-second MPC simulation. We perform a Monte-Carlo analysis of the algorithm by sampling the initial state of the system as well as the objective parameter θ. The aggregated results from 50 samples are presented in Figure 4. They show a significant decrease of the parameter estimation error.

C. Trajectory Prediction

Additionally, we show that LUCIDGames allows the robot to better predict the trajectory of the other agents. We use the same Monte Carlo analysis as described above on three driving scenario (Fig. 5). First, we observe that LUCIDGames initialized with a coarse prior starts with a large prediction error; around 4 meters in all scenarios. However, it converges to a prediction error very much comparable to the one obtained with the oracle in about 1 s. This illustrates the ability of the robot using LUCIDGames to quickly improve its predictions about the surrounding agents by gathering information about them. The error obtained by keeping the coarse prior remains high and fairly constant during the simulation. The one obtained using LUCIDGames is an order of magnitude lower, in comparison, by the end of the simulation.

Second, we compare LUCIDGames to a non-game-theoretic baseline on trajectory prediction error. This baseline predicts the trajectories of the agents surrounding the robot by propagating straight-line and constant-speed trajectories for each agent. These predicted trajectories are of the same duration (3 seconds) as the open-loop predictions made by LUCIDGames. This line-prediction baseline may seem very coarse. However, in the context of highway driving, straight line trajectories are very pertinent for short (3 seconds) horizon predictions. In practice, we use a straight highway environment for our simulations (Fig. 2). As the roadway is not curved, the only causes of trajectory curvature are lane changes, nudging and merging maneuvers. LUCIDGames is able to outperform the baseline by capturing these natural driving behaviors that go beyond lane following.

For the overtaking scenario (Figure 5), LUCIDGames starts off with a large prediction error but quickly converges to prediction error lower than the line-prediction baseline. However, the performance gap is small confirming that the line prediction baseline is a suitable model for short horizon prediction in typical highway driving.

On the other hand, for more complex driving scenarios like ramp merging, the gap between LUCIDGames and the line prediction technique is significant. We observe that this gap is the highest after 1 s when LUCIDGames has successfully converged. The gap consistently decreases afterwards as the system converges to a steady state where all the vehicles drive in straight lines following their desired lanes.

Similarly, for the collision-avoidance scenario, the prediction error obtained using LUCIDGames is around half that obtained using the line-prediction baseline after the parameter estimation has converged (1 s). We observe that the line prediction baseline almost matches LUCIDGames' when the vehicles are constrained to drive on a narrower roadway ($t \in [3, 4]$s). Finally, after the obstacle is passed ($t \in [4, 6]$s), the performance gap increases in favor of LUCIDGames. Indeed, it is able to predict that vehicles are going to return to their desired lanes after avoiding the obstacle.

D. Safe Trajectory Planning

We implement a robust trajectory planning scheme for the robot that accounts for uncertainty in the objective of the other agents by enforcing "safety constraints." With LUCIDGames, we maintain a Gaussian belief over the other agents' objectives. We thus quantify the uncertainty of our current objective function estimates. Taking into account such uncertainty can be instrumental in preventing the robot from making unsafe decisions. For instance, an autonomous vehicle should act cautiously when overtaking an agent for which it has an uncertain estimate of its desired speed and desired lane.

For many multi-robot systems, safety is ensured by avoiding collisions with other agents. Thus, we encode safe decision making for the robot by ensuring its decisions are robust to misestimation of the objective functions. First, the robot computes the "safety constraints," which are inflated collision avoidance constraints around other agents by fitting ellipses around the trajectories sampled by the UKF (e.g., in the 95%-confidence ellipse). These safety constraints can be seen as approximate chance constraints that can be efficiently computed. Then, the robot solves the dynamic game corresponding to the mean of the belief over θ, with the safety constraints. In practice, when the uncertainty about θ is large, the sampled sigma-points and their corresponding trajectories are scattered and generate a large collision avoidance zone. The top roadway visualization in Figure 1 illustrates this situation. These safety constraints can be seen as a lifting of the uncertainty in the low-dimensional space of objective parameters onto the high-dimensional space of predicted trajectories.

We showcase the driving strategy emerging from this robust planning scheme in Figure 1. The human driver and the robot are confronted with an obstacle. Using LUCIDGames, the robot infers the human's intent to change lanes (to avoid the obstacle), and negotiates, through the game theoretic planner, whether to yield to the human, or to let the human yield. In phase 1, the robot has a large initial uncertainty about the objective of the human-driven vehicle (blue). Indeed, the set of sampled trajectories contains both predictions where the human cuts in front of the robot, and ones in which the human yields to let the robot go first. Thus, the robot slows down to comply with the safety constraints which covers both hypotheses. In phase 2, the robot has correctly estimated the human's intent to yield to the robot, to change lanes after the robot passes. The collision avoidance zone generated by the safety constraints shrinks. This allows the robot to plan an overtaking maneuver and regain speed, while the human changes lanes behind the robot to avoid the obstacle (phase 3). LUCIDGames without these safety constraints does not slow down to account for the initial uncertainty. The same is true for the oracle and the straight line prediction baseline.

E. Results Discussion

1) Parameter Estimation: We have observed challenging scenarios demonstrating the complexity of the objective estimation task. For instance, if all the agents are far from each other, none of the collision avoidance penalties (14) are "active". In such a situation, it is impossible to estimate the aggressiveness parameter that scales the cost of these collision avoidance penalties. Nevertheless, we argue that this observability issue is not crucial in practice. Indeed, as long as the agents remain far from each other the aggressiveness parameter will not affect the trajectory prediction of the robot.

2) Trajectory Prediction: The human-driven vehicles in these simulations are modeled as agents solving the ground-truth dynamic game for a Nash equilibrium strategy in a receding horizon loop. We notice in Figure 5 that even when the robot has access to the ground-truth objective functions, its trajectory prediction error is not null because of the noise added to the dynamics. LUCIDGames consistently converges to error levels similar to the ones of the agent having access to the ground-truth objective functions.

VII. Conclusion

We have presented LUCIDGames, a game theoretic planning framework that includes the solution of an inverse optimal control problem online to estimate the objective function parameters of other agents. We demonstrate that this algorithm is fast enough to run online in a receding-horizon loop, and is effective in planning for an autonomous vehicle to negotiate complex driving scenarios while interacting with other vehicles. We showed that this method outperforms two benchmark planning algorithms, one assuming straight-line predictions for other agents, and one incorporating game-theoretic planning, but without online parameter estimation of other agents' objective functions.

We envision several promising directions for future work: In this work, the set of objective function parameters has been designed with "expert" knowledge of the problem at hand, so that they encompass a large diversity of driving behaviors while remaining low dimensional. However, one could envision these parameters and associated features being identified via a data-driven approach. The overall approach of estimating online a reduced set of parameters to better predict the behavior of the system is appealing. Indeed, in this framework, the dynamic game solver lifts the low dimensional space of objective function parameters (order 10^1) into the high dimensional space of predicted trajectories (order 10^2 - 10^3). This lifting or "generative model" natively embeds safety requirements by generating dynamically feasible trajectories respecting collision avoidance constraints. It also accounts for the fact that agents tend to act optimally with respect to some objective functions. Finally, through its game-theoretic nature, it captures the reactive nature of the agents surrounding the robot in autonomous driving scenarios, where negotiation between players is a crucial feature.

References

[1] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Systems, (Taipei)*, Oct. 2010, pp. 797–803.

[2] S. L. Cleac'h, M. Schwager, and Z. Manchester, "ALGAMES: A fast solver for constrained dynamic games," in *Robot.: Sci. Syst. XVI, Robot.: Sci. Syst. Found.*, Jul. 2020, doi: 10.15607/RSS.2020.XVI.091.

[3] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, "Efficient Iterative Linear-Quadratic Approximations for Nonlinear Multi-Player General-Sum Differential Games," *IEEE Int. Conf. Robot. Autom. (ICRA)*, Mar. 2020, pp. 1475–1481.

[4] J. Colyar and J. Halkias, "US highway 101 dataset," Tech. Rep. FHWA-HRT-07-030, Federal Highway Administration (FHWA), 2007.

[5] A. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in Proc. Icml, vol. 1, p. 2, 2000.

[6] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," *Assoc. Advance. Artif. Intell.* , vol. 8, pp. 1433–1438, 2008.

[7] K. Waugh, B. D. Ziebart, and J. A. Bagnell, "Computational rationalization: *The Inverse Equilibrium Problem*," Aug. 2013, *arXiv:1308.3506*.

[8] R. Spica, D. Falanga, E. Cristofalo, E. Montijano, D. Scaramuzza, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing," *IEEE Trans. Robot.*, vol. 26, no. 5, pp. 1389–1403, 2020.

[9] T. Basar and G. J. Olsder, Dynamic noncooperative game theory, vol. 23, 1999.

[10] H. Chen, R. Ye, X. Wang, and R. Lu, "Cooperative control of power system load and frequency by using differential games," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 3, pp. 882–897, May 2015.

[11] N. Ratliff, Learning to search: Structured prediction techniques for imitation learning. Ph.D. thesis, Pittsburg, PA: Carnegie Mellon University, May 2009.

[12] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotionan inverse optimal control approach," *Auton. Robots*, vol. 28, pp. 369–383, Apr. 2010.

[13] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, pp. 3916–3924, 2016.

[14] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "Modeling interaction via the principle of maximum causal entropy," *Proce. 27th Int. Conf. Int. Conf. Mach. Learn.*, pp. 1255–1262, 2010.

[15] J. Inga, E. Bischoff, F. Köpf, M. Flad, and S. Hohmann, "Inverse cooperative and non-cooperative dynamic games based on maximum entropy inverse reinforcement learning," Nov. 2019, *arXiv:1911.07503*.

[16] T. L. Molloy, J. Inga, M. Flad, J. J. Ford, T. Perez, and S. Hohmann, "Inverse open-loop noncooperative differential games and inverse optimal control," *IEEE Trans. Autom. Control*, vol. 65, no. 2, pp. 897–904, Feb. 2020.

[17] R. Van der Merwe and E. Wan, "The square-root unscented kalman filter for state and parameter-estimation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. Proc.*, 2001, vol. 6, pp. 3461–3464.

[18] S. Hoermann, D. Stumper, and K. Dietmayer, "Probabilistic long-term prediction for autonomous vehicles," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2017, pp. 237–243.

[19] R. Bhattacharyya, R. Senanayake, K. Brown, and M. Kochenderfer, "Online parameter estimation for human driver behavior prediction," *Amer. Cont. Conf. (ACC)*, 2020, pp. 301–306.

[20] J. Schulz, C. Hubmann, J. Lochner, and D. Burschka, "Multiple model unscented kalman filtering in dynamic bayesian networks for intention estimation and trajectory prediction," in *Proc. 21st Int. Conf. Intell. Transp. Syst.*, Nov. 2018, pp. 1467–1474.

[21] Z. N. Sunberg, C. J. Ho, and M. J. Kochenderfer, "The value of inferring the internal state of traffic participants for autonomous freeway driving," in *Proc. IEEE Amer. Control Conf.*, May 2017, pp. 3004–3010.

[22] Z. Sunberg and M. Kochenderfer, "Improving automated driving through planning with human internal states," May 2020, *arXiv:2005.14549*.

[23] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proc. Nat. Acad. Sci.*, vol. 116, pp. 24972–24978, Dec. 2019.

[24] E. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proc. IEEE Adaptive Syst. Signal Process., Commun., Control Symp. (Cat. No.00EX373)*, (Lake Louise, Alta., Canada), 2000, pp. 153–158.

[25] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction," in *Proc. IEEE Int. Conf. Robot. Automat.*, (Brisbane, QLD), May 2018, pp. 3399–3406.

[26] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron: Dynamically-feasible trajectory forecasting with heterogeneous data," Jun. 2020, *arXiv:2001.03093*.

[27] M. Bansal, A. Krizhevsky, and A. Ogale, "ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst," *Proc. Robot.: Sci. Syst.*, 2019, Jun. 2018, doi: 10.15607/RSS.2019.XV.031.

[28] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Herbert, "Activity forecasting," in *Proc. Eur. Conf. Comput. Vis.*, (Berlin, Heidelberg), Springer, 2012, pp. 201–214.

[29] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), (Madrid)*, Oct. 2018, pp. 1534–1539.

[30] Aurora, "Safety report," Tech. Rep., Aurora Innovation, 2019.

[31] M. Bouton, J. Karlsson, A. Nakhaei, K. Fujimura, M. J. Kochenderfer, and J. Tumova, "Reinforcement learning with probabilistic guarantees for autonomous driving," May 2019, *arXiv:1904.07189*.

[32] F. Facchinei and C. Kanzow, "Generalized nash equilibrium problems," 4OR, vol. 5, pp. 173–210, Sep. 2007.