LUCIDGames: onLine UnsCented Inverse Dynamic Games for Adaptive Trajectory Prediction and Planning.

Simon Le Cleac'h[1], Mac Schwager[2] and Zachary Manchester[3]

***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 paper 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`.**

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 crowd navigation problem, the robot might know the preferred walking speed of humans. 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] or the ETH dataset [5] for interacting pedestrians. Inverse reinforcement learning (IRL) approaches typically learn a general objective function to suit a large batch of demonstrations from multiple agents [6], [7], [8]. 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.

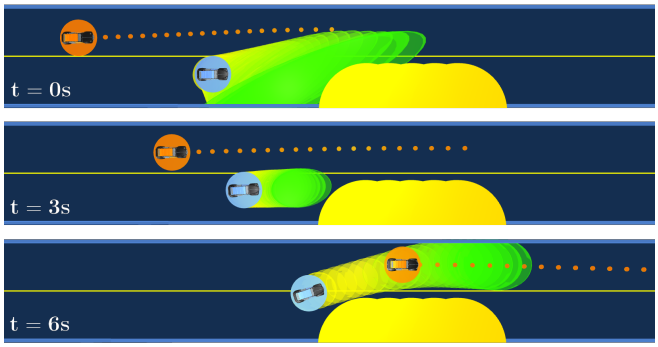

Fig. 1. We present three roadway visualizations of a scenario where the autonomous vehicle (AV) in orange and a human-driven car (blue) have to overtake a large obstacle (yellow) on the bottom lane. The AV (orange) follows the robust version of LUCIDGames. At the start, the AV slows down to avoid the 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.

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

This work was supported in part by NSF NRI grant 1830402 and DARPA YFA grant D18AP00064. 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.

[1]Simon Le Cleac'h is with the Department of Mechanical Engineering, Stanford University, California, USA `simonlc@stanford.edu`

[2]Mac Schwager is with the Department of Aeronautics & Astronautics, Stanford University, California, USA `schwager@stanford.edu`

[3]Zachary Manchester is with the Robotics Institute, Carnegie Mellon University, Pennsylvania, USA `zacm@cmu.edu`

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. 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 40Hz 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.

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, we show that LUCIDGames with safety constraints allows for realistic, cautious interactions between a robot and another agent making a 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 [3], [2], power system control [9], drone and car racing [10] etc. The solutions of a dynamic game depend on the type of equilibrium selected [11]. 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], [10], [9]. They seem to capture the game-theoretic interactions observed in some multi-agent non-cooperative problems, e.g. ramp merging. 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 [7]. 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 [12], [13]. However, these works are limited to single-agent problems.

In the multi-agent setting, some IRL approaches formulate the problem by assuming cooperative agents [14] or competing agents [8], [15]. 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 [16], [17]. 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. In the multi-agent setting, several approaches were proposed to solve this type of online parameter estimation problem.

C. Online Parameter Estimation

We choose the multi-agent autonomous driving problem of highway driving as our running example throughout this paper. We review approaches proposed for online parameter estimation in this context. The Intelligent Driver Model (IDM) [18], is a rule-based lane-following model balancing two objectives: reaching a desired speed and avoiding collision with the preceding car. This model has few tunable parameters and returns the scalar acceleration of the vehicle along a predefined path. Several works used online filtering techniques to estimate these IDM parameters [19], [20]. Schultz et al. used particle filtering techniques to estimate the driver's objective including discrete decision variables like turning left or right [21]. These works demonstrated that estimating the surrounding drivers objectives helps better predict their future trajectories.

However, this gained information was not used to improve the decision making of the cars. A recent work estimated the social value orientation (SVO) of the agent the robot is interacting with [22]. However, this formulation still requires knowledge of the desired lane or desired speed of each agent. This is precisely the assumption we want to avoid in the present work.

D. Data-Driven Trajectory Prediction

There is a rich literature on applying data-driven approaches to pedestrian or vehicle trajectory predictions. Such approaches usually require a large corpus of data and are trained offline as a general model to suit multiple agents. On the other hand, we require a small amount of data and find parameters online for a specific agent. Data-driven methods can predict a distribution over future trajectories e.g., offline inverse optimal control with online goal inference [23]; Conditional Variational Autoencoders (CVAE) [24] or Generative Adversarial Networks (GAN) [25]. 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 [26], could prevent infeasible trajectory predictions [27]. Our approach generates dynamically feasible and collision-free predictions. One notable work in this field is Trajectron++ [28]. 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 [20] with a learning-based method [25].

III. Problem Statement

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 necessary 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 [19], [20], [22], but this method ignores the reactive nature of human drivers to the robots' decisions. Lane-following models, such as IDM [29] fail to capture complex driving strategies like nudging or changing lanes that our model can generate. Moreover, 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 experiments in this paper are run with the "ideal" agents having noisy estimates of the objectives of the surrounding agents in the scene. 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.

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

A. 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 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,

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

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. We compactly express these constraints as a set of inequalities $C : \mathbb{R}^{\bar{n}+\bar{m}} \mapsto \mathbb{R}^{n_c}$:

$$\begin{aligned} \min_{X, U^\nu} \quad & J^\nu(X, U^\nu), \\ \text{s.t.} \quad & C(X, U) \leq 0. \end{aligned} \tag{2}$$

Finding a Nash-equilibrium solution to the set of M Problems (2) is called a generalized Nash equilibrium problem (GNEP) [2], [30]. 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 (2) 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 initial state of the system and the objective functions, $J^1,\dots,J^M$, and returns an open-loop trajectory of the joint system comprising the robot and the ideal agents.

B. Objective Function Parameterization

As is typically the case in the IRL and IOC literature [7], [13], [16], 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,

$$J^\nu(X, U^\nu) = \phi(X, U^\nu)^T \theta^\nu. \tag{3}$$

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,

$$\theta = [{\theta^1}^T \dots \ {\theta^{r-1}}^T, {\theta^{r+1}}^T \dots \ {\theta^M}^T]^T \in \mathbb{R}^q. \tag{4}$$

IV. Unscented Kalman Filtering Formulation

We propose an algorithm that 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 Unscented Kalman Filter (UKF) [31] 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. We assume this quantity evolves according to a random walk, so the process model for the UKF is the identity map plus Gaussian white process noise. The crucial part of our algorithm is the measurement model. In our case, the measured quantity available to the robot (with noise) is the 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. Therefore, our UKF requires the solution of the dynamic game for each sigma-point of θ at each time step. The dynamics and measurement models, as well as the steps in our UKF estimator are described in detail below.

A. Process and 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 assume that the ground-truth parameter θ is a random walk with relatively small process noise covariance, which means that the agents' objectives are nearly constant, but may change slightly over the course of the estimation. This is a reasonable assumption as, for many robotics applications, an agent's objective corresponds to its long-term goal and thus varies over time scales far larger than the estimator's update period. The process model corresponds to an additive white Gaussian noise and is defined as follows,

$$\theta_{t+1} = \theta_t + \delta_t, \qquad \delta_t \sim \mathcal{N}(0, Q_t). \tag{5}$$

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]:

$$x_t = g(\theta, x_{t-1}) + \varepsilon_t, \tag{6}$$

$$\varepsilon_t \sim \mathcal{N}(0, R). \tag{7}$$

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.

[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 1 Parameter estimation module.

1: **procedure** ESTIMATOR(x_{t-1}, x_t, μ_{t-1}, μ_t, Σ_t)
2: $\bar{\mu}_{t+1} \leftarrow \mu_t$
3: $\bar{\Sigma}_{t+1} \leftarrow \Sigma_t + Q_t$
4: $\Theta, M, C \leftarrow$ SIGMAPOINTS($\bar{\mu}_{t+1}, \bar{\Sigma}_{t+1}$) ▷ Eq. 8-14
5: $\chi^{(i)} \leftarrow$ CONTROLLER($x_{t-1}, \mu_{t-1}, \Theta^{(i)}$) $\forall i$
6: $\bar{x}_t \leftarrow \sum_i M^{(i)} \chi^{(i)}$
7: $P \leftarrow \sum_i C^{(i)} [\chi^{(i)} - \bar{x}_t][\chi^{(i)} - \bar{x}_t]^T$
8: $S \leftarrow \sum_i C^{(i)} [\Theta^{(i)} - \bar{\mu}_{t+1}][\chi^{(i)} - \bar{x}_t]^T$
9: $K \leftarrow SP^{-1}$
10: $\mu_{t+1} \leftarrow \bar{\mu}_{t+1} + K(\chi - \bar{x}_t)$
11: $\Sigma_{t+1} \leftarrow \bar{\Sigma}_{t+1} - KPK^T$
12: **return** μ_{t+1}, Σ_{t+1}

Algorithm 2 Decision making process of the agents.

procedure CONTROLLER(x_t, μ_t, θ)
$U_t^r \leftarrow$ ALGAMES(x_t, μ_t) ▷ Robot's plan
$U_t^{-r} \leftarrow$ ALGAMES(x_t, θ) ▷ Ideal agents' plans
$U_t \leftarrow [U_t^{rT}, U_t^{-rT}]^T$
$x_{t+1} \leftarrow$ DYNAMICS(x_t, U_t) ▷ Equation 1
return x_{t+1}

B. UKF Algorithm

The estimator propagates the belief over the vector θ in time. The procedure that updates this belief is described in Algorithm 1. Lines 2 and 3 correspond to the prediction step, which exploits the process model. Line 4 samples sigma-points from a Gaussian distribution over the vector θ (Eq. 8-14),

$$\lambda = \alpha^2(q+\kappa) - q, \tag{8}$$

$$\Theta^{(0)} = \mu, \tag{9}$$

$$\Theta^{(i)} = \mu + (\sqrt{(q+\lambda)\Sigma})_i, \qquad \forall i \in \{1,\dots,q\} \tag{10}$$

$$\Theta^{(i)} = \mu - (\sqrt{(q+\lambda)\Sigma})_{i-q}, \qquad \forall i \in \{q+1,\dots,2q\} \tag{11}$$

$$M^{(0)} = \lambda/(q+\lambda), \tag{12}$$

$$C^{(0)} = \lambda/(q+\lambda) + (1-\alpha^2+\beta), \tag{13}$$

$$M^{(i)} = C^{(i)} = \lambda/\big(2(q+\lambda)\big). \qquad \forall i \in \{1,\dots,2q\} \tag{14}$$

We follow a classical sampling scheme that relies on parameters, α, β, κ, controlling the spread of the sigma-points and encoding prior knowledge about the distribution. Wan et al. provide a detailed interpretation for these parameters [31]. Line 5 applies the measurement model to the sampled sigma-points, Θ, which are guesses over the vector θ. Specifically, for each sigma-point, we solve the dynamic games that the ideal agents and the robot encountered at the previous time step. Then, we propagate the system's dynamics for one time step to obtain χ, a set of predictions over x_t, the current state of the system. Lines 6 to 9 compute the Kalman gain, K, and measurement prediction $\bar{x}_t$. Finally, the update step is executed in lines 10 and 11.

C. 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

Algorithm 3 Combined estimator and planning module.

procedure LUCIDGAMES(x_{t-1}, x_t, μ_{t-1}, μ_t, Σ_t)
  for $t = 1, 2, \ldots$ **do**
    $x_{t+1} \leftarrow$ CONTROLLER(x_t, μ_t, θ)
    $\mu_{t+1}, \Sigma_{t+1} \leftarrow$ ESTIMATOR($x_{t-1}, x_t, \mu_{t-1}, \mu_t, \Sigma_t$)
  return $x_{t+1}, \mu_{t+1}, \Sigma_{t+1}$

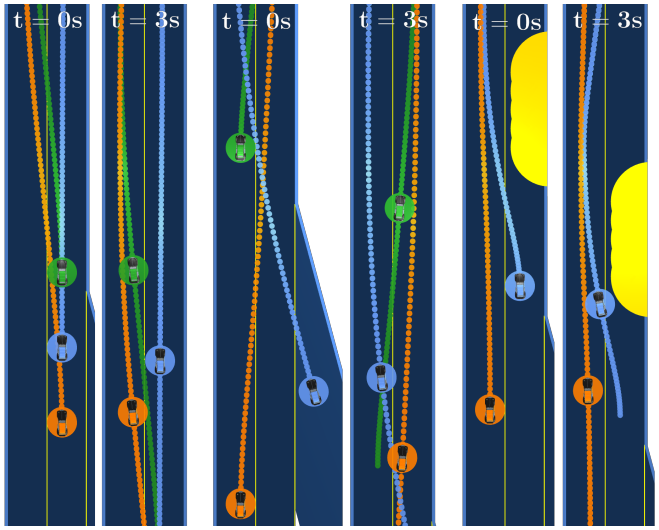

Fig. 2. From left to right, we represent three highway driving scenarios, an overtaking maneuver, a ramp merging, and an obstacle avoidance maneuver. For each scenario we represent the system at $t = 0$s and $t = 3$s.

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 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 problems involving a high level of interactions between agents. Specifically, we test LUCIDGames in three driving scenarios exhibiting maneuvers such as overtaking, ramp merging and obstacle avoidance (Figure 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,

$$J^\nu(X, U^\nu) = \sum_{k=1}^{N-1} \frac{1}{2}(x_k - x_f)^T Q (x_k - x_f) + \frac{1}{2} u_k^{\nu T} R u_k^\nu + \frac{1}{2}(x_N - x_f)^T Q_f (x_N - x_f) + \sum_{k=1}^{N} \sum_{\mu \neq \nu} \gamma^\nu \left(\max\left(0, ||p_k^\nu - p_k^\mu||_2 - \eta(r^\nu + r^\mu)\right)\right)^2. \quad (15)$$

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. They correspond to 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. We remark that this parameterization is consistent with an objective function expressed as a linear combination of features, as in Equation 3. 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

To assess the merits of LUCIDGames, we test it on highway driving scenarios as shown on Figure 2. We first assess the tractability and scalability of the approach for an increasing number of agents. 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. [20] used highway driving datasets to compare the trajectory prediction performance of rule-based and black-box driver models. The set of evaluated methods included among others: constant velocity prediction, Generative Adversarial Imitation Learning (GAIL) [25] 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:

$$\mu_0 = \mathbf{1}, \qquad \Sigma_0 = v_0 I_q. \quad (16)$$

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 (40Hz) and up to four agents (10Hz). 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

# 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.

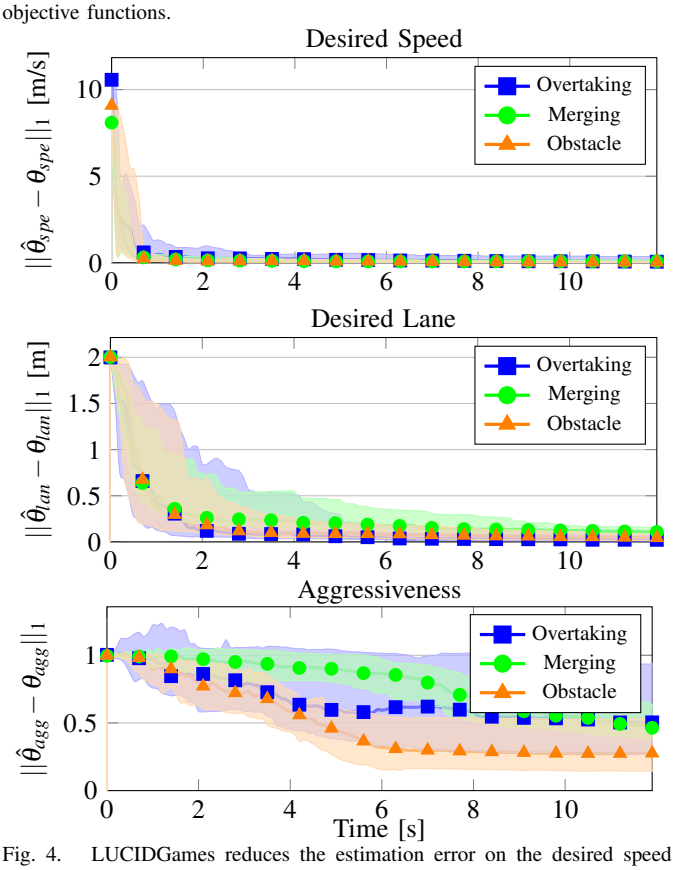

Fig. 4. LUCIDGames reduces the estimation error on the desired speed parameter by a factor of 100 within 12 seconds of interaction (top plot). The error on the desired lane is divided by 20 (middle plot) and the error on the aggressiveness parameter is halved (bottom plot). The markers indicate the median error computed over 50 simulations. The faded color areas correspond to the interval between the 1^{st} and 3^{rd} quantile.

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. We test LUCIDGames 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. We observe a significant decrease in the relative error between the ground-truth parameter and the mean of the belief.

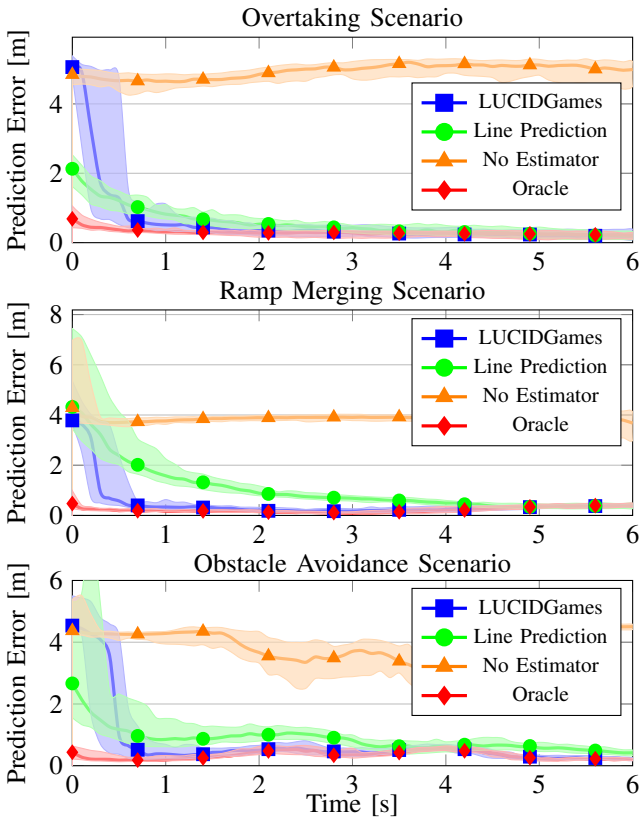

Fig. 5. We present the trajectory prediction error obtained by the robot on 3 scenarios for 4 algorithms: LUCIDGames; a non-game-theoretic baseline using straight line predictions; a game-theoretic solver that does not estimate the other agents' objective functions and finally, an oracle having access 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 1^{st} and 3^{rd} quantile.

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 m in all scenarios. However, it converges to a prediction error very much comparable to the one obtained with the oracle in about 1 second. 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. We choose to compare our approach to this baseline to assess

the impact of including game-theoretic reasoning on the trajectory prediction performance. 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 (Figure 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 second 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 second). 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. The "keep-out" zone is cone-shaped in free space as expected but shrinks down when the roadway narrows or when the sampled trajectories concur towards the same position.

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. Since the robot's estimate of the human's objective is more certain, the collision avoidance zone generated by the safety constraints shrinks. This allows the robot to plan an overtaking maneuver and regain speed. In phase 3, the robot safely proceeds in its own lane cruising at its desired velocity, while the human changes lanes behind the robot to avoid the obstacle. On the other hand, 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. We observe the same behaviors on the scenario where the robot yields to the human.

E. Results Discussion

1) Tractability: We demonstrate the tractability of the algorithm for up to four agents with an update rate of 10Hz. However, it is important to notice that the robot's controller and its estimator can be run at different rates. This would allow for a fast update of the robot's plan and a slower update of its estimation of the objective functions of the other agents.

2) Parameter Estimation: The good estimation performance of LUCIDGames was assessed through a Monte Carlo analysis. However, 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 (Eq. 15) 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. Conversely, as soon as the agents get closer to each other, the aggressiveness parameter matters but the observability issue disappears.

3) 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. This is due the noise added to the dynamics as well as the discrepancy between the open-loop plan predictions and the actual trajectory stemming from executing open-loop strategies in a receding horizon loop. We observe in Figure 5 that LUCIDGames consistently converges to error levels closely matching 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 *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Taipei), pp. 797–803, IEEE, Oct. 2010.

[2] S. Le Cleac'h, M. Schwager, and Z. Manchester, "ALGAMES: A Fast Solver for Constrained Dynamic Games," in *Robotics: Science and Systems XVI*, Robotics: Science and Systems Foundation, July 2020.

[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," *arXiv:1909.04694 [cs, eess]*, Mar. 2020.

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

[5] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th International Conference on Computer Vision*, (Kyoto), pp. 261–268, IEEE, Sept. 2009.

[6] A. Ng and S. Russell, "Algorithms for Inverse Reinforcement Learning," in *Icml*, 2000.

[7] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," *Association for the Advancement of Artificial Intelligence*, p. 6, 2008.

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

[9] H. Chen, R. Ye, X. Wang, and R. Lu, "Cooperative Control of Power System Load and Frequency by Using Differential Games," *IEEE Transactions on Control Systems Technology*, vol. 23, pp. 882–897, May 2015.

[10] 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," *arXiv:1801.02302 [cs]*, Jan. 2018.

[11] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. Society for Industrial and Applied Mathematics, 1998.

[12] N. Ratliff, *Learning to Search: Structured Prediction Techniques for Imitation Learning*. PhD thesis, Carnegie Mellon University, Pittsburg, PA, May 2009.

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

[14] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative Inverse Reinforcement Learning," *Advances in neural information processing systems*, p. 9, 2016.

[15] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "Modeling Interaction via the Principle of Maximum Causal Entropy," *Journal Contribution*, p. 8, 2010.

[16] 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," *arXiv:1911.07503 [cs, eess]*, Nov. 2019.

[17] 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 Transactions on Automatic Control*, vol. 65, pp. 897–904, Feb. 2020.

[18] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, pp. 1805–1824, Aug. 2000.

[19] S. Hoermann, D. Stumper, and K. Dietmayer, "Probabilistic long-term prediction for autonomous vehicles," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, (Los Angeles, CA, USA), pp. 237–243, IEEE, June 2017.

[20] R. Bhattacharyya, R. Senanayake, K. Brown, and M. Kochenderfer, "Online Parameter Estimation for Human Driver Behavior Prediction," *arXiv:2005.02597 [cs]*, May 2020.

[21] 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 *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, (Maui, HI), pp. 1467–1474, IEEE, Nov. 2018.

[22] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proceedings of the National Academy of Sciences*, vol. 116, pp. 24972–24978, Dec. 2019.

[23] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Herbert, "Activity Forecasting," in *European Conference on Computer Vision*, (Berlin, Heidelberg), Springer, 2012.

[24] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, (Brisbane, QLD), pp. 3399–3406, IEEE, May 2018.

[25] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-Agent Imitation Learning for Driving Simulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Madrid), pp. 1534–1539, IEEE, Oct. 2018.

[26] Aurora, "Aurora: Safety Report (2019)," tech. rep., Aurora Innovation, 2019.

[27] M. Bansal, A. Krizhevsky, and A. Ogale, "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst," *arXiv:1812.03079 [cs]*, Dec. 2018.

[28] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data," *arXiv:2001.03093 [cs]*, June 2020.

[29] M. Bouton, J. Karlsson, A. Nakhaei, K. Fujimura, M. J. Kochenderfer, and J. Tumova, "Reinforcement Learning with Probabilistic Guarantees for Autonomous Driving," *arXiv:1904.07189 [cs]*, May 2019.

[30] F. Facchinei and C. Kanzow, "Generalized Nash equilibrium problems," *4OR*, vol. 5, pp. 173–210, Sept. 2007.

[31] E. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, (Lake Louise, Alta., Canada), pp. 153–158, IEEE, 2000.