Learned Objectives for Game Theoretic Planning

Keiko Nagami*[1], Jaden Clark*[2], Mac Schwager[1]

***Abstract*— Modeling multi-agent motion planning problems as dynamic games allows agents in the system to leverage the objectives of non-cooperative agents in their own planning. Using this approach has been shown to improve performance over methods that do not solve dynamic games in non-cooperative, competitive examples such as lane merging and racing. Unfortunately, many existing game theoretic planning algorithms can require long computation times, preventing real world implementation. Furthermore, most algorithms assume knowledge of the objective functions of other agents in the system. Methods that do not make this assumption will often assume the structure of the objective function, and assign parameters that best represent the agent's behavior. However, this approach can be limiting in cases where the underlying objective does not match the assumed form. To address these challenges, we propose a method that uses a neural network to predict trajectories of players in a game. This output trajectory is then used to define a tracking cost that represents the objective of each player in the game. From here, traditional model-based game theoretic planning approaches can be used to compute the Nash equilibrium trajectories for all players in the system. We demonstrate our approach in a racetrack simulation environment, and show that our learning-based method is able to closely match the trajectories formed in the case where the ground truth objectives are known.**

I. INTRODUCTION

In many real-world applications of autonomous systems such as self-driving cars, mobile robot navigation in crowds, and autonomous racing, the interactions among agents play a crucial role. Specifically, the autonomous agents deployed in the real world must interact with other non-cooperative or competitive agents, without direct communication.

Traditional planning approaches often adopt a predict-then-plan architecture, where other agents' trajectories are predicted and treated as fixed obstacles in the planning process. This method can be effective in cases where other agents in the system are largely static, far from the ego agent, or relatively slow moving. However, this approach may be insufficient when the other agents in the system are reacting to actions of the ego agent, navigating in close proximity, and even trying to prevent the ego agent from reaching its goals. Disregarding the influence of the robot's own trajectory on other agents and can lead to suboptimal or even unsafe plans, as it fails to capture the reactive nature of the agents in the environment. In such cases, game theoretic planning offers a valuable framework for identifying solution methods to these complex decision-making problems.

*Authors contributed equally

[1]Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA, {knagami, schwager}@stanford.edu

[2]Department of Computer Science, Stanford University, Stanford, CA 94305, USA, jvclark@stanford.edu

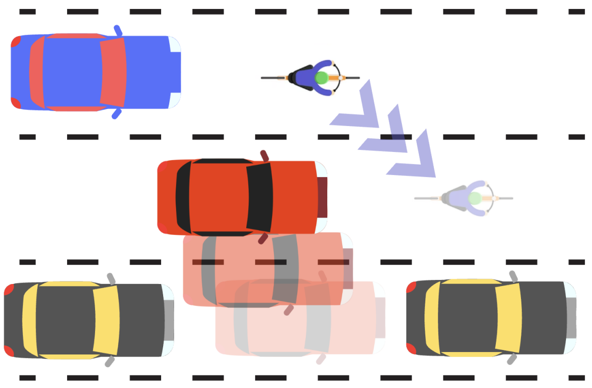

Fig. 1. Autonomous vehicles deployed in the real-world must be robust to complex interaction with non-cooperative agents in the system.

Game theoretic planning provides a more comprehensive and dynamic approach by considering all agents as players in a game. This way, agents can strategically reason about their decisions, taking into account the potential strategic reactions of other agents. In this paper, we approach the task of solving for a Nash equilibrium, where no player can unilaterally improve their outcome. While game theoretic planning shows promise for addressing multi-agent planning problems, it also presents challenges. Dynamic games are inherently more complex to solve than single-agent optimization problems due to the involvement of multiple players with potentially competing objectives. Furthermore, finding equilibrium solutions in these games is not guaranteed, even in simple scenarios [1]. This nonexistence of equilibria adds to the complexity of solving dynamic games.

In solving for Nash equilibria of multiple agents in a game, a key component is the objective of each player in the system. Many existing methods assume that the objective function of each player is known. However, this assumption may not always hold. Methods to estimate the objective exist but enforce a structure to the objective function, which may not adequately represent the underlying function.

An alternative approach to using game theoretic planning methods is to use learning-based methods. Imitation learning and reinforcement learning techniques have become increasingly common approaches to solving robotic motion planning problems [1]. Learning-based approaches, have gained significant interest in autonomous driving due to their potential to handle the complexity of interactions in dense urban environments with small online computational complexity. Traditional motion planning methods often struggle in such scenarios, as they require defining complex cost functions that align with human expectations. Imitation learning, on the other hand, leverages expert demonstrations to learn either a cost function or a policy directly, making it well-suited for handling diverse driving situations.

One key benefit of learning-based approaches is the reduction in manual parameter tuning. Instead of iteratively tuning each parameter of a cost function while trying to maintain performance in all foreseeable scenarios, deep learning algorithms can learn appropriate parameters from data. This allows the system to adapt to different driving conditions without extensive manual tuning. Another advantage is the potential for end-to-end learning. In some approaches, the system learns to map observations directly to low-level vehicle control interface commands, eliminating the need for explicit modeling or hand-engineered features [3]. This can simplify the overall system architecture and potentially lead to more robust and adaptable control. However, evaluating such policies is challenging. While model-based approaches explicitly enforce constraints on the trajectory they produce, constraining neural network outputs is more challenging. Particularly in the case of distribution shift, when the driving scenario deviates significantly from those demonstrated by the expert, learned driving policies may produce dangerous or dynamically infeasible control actions [4].

In this paper, we present a method that fuses both imitation learning and game-theoretic trajectory optimization. In this work, we consider the case where the opponent's objectives are not known apriori and must be learned offline through interactions with the ego agent. By learning from an agent planning over a longer horizon, we will be able to instill longer horizon planning information into the planner for the learning agent, even if this agent plans a shorter trajectory. This decreases online runtime complexity of our algorithm. Furthermore, we will simultaneously learn to predict trajectories of other agents. By predicting trajectories as opposed to parameters of an objective function as typically done in inverse reinforcement learning, the agent can predict paths of other agents with arbitrarily structured objectives, thus eliminating much of the cost-function fine-tuning associated with model-based trajectory optimization.

The main contributions of this work are as follows:

- A formalized game theoretic framework to ground neural network policy-generated waypoints with trajectory optimization
- Demonstrate that learning from agents using long-horizon planning can decrease online planning horizon for agents executing game-theoretic MPC online
- Show that a prediction network that outputs multi-agent trajectories can be used in place of a fixed "goal state" for objective of agents

II. RELATED WORKS

A. Game-Theoretic Approaches

Recent work in game-theoretic planning have explored various approaches for equilibrium selection. One class of equilibrium selection is the Stackelberg equilibrium, which assumes asymmetry between players. Several studies have focused on searching for Stackelberg equilibria [5], [6], where a leader player chooses a strategy first, and a follower player responds accordingly. However, this approach can lead to overly aggressive or passive behavior due to asymmetry [1]. Alternatively, Nash equilibria assume symmetry between players. The Nash equilibrium strategy is defined as the set trajectories where each player's strategy is unilaterally the best response. In other words, any given agent's objective can not be improved without changing the strategy of other agents. Nash equilibria are more suitable for symmetric, multi-player interactions and have shown to produce more natural behavior compared to Stackelberg equilibria when solving for open-loop strategies.

In solving for Nash equilibria, a common approach is to use iterative best response. In these algorithms, the solver iteratively optimizes for each player's objective, while holding the decision variables of the other players constant, as done in [7], [8], [9], and [10]. These approaches are easy to interpret and scale reasonably well with the number of agents in the system. They also do not require the cost function to be quadratic. However, convergence of these algorithms is not properly understood, and can have long computation times. Other approaches enforce a quadratic objective function for all players in the game, and are able to generate solutions for real time planning [2], [11], [12].

B. Inverse Solutions

In this work, we focus on problems where the objectives of other players in the game are unknown. Objective function estimation methods, such as Inverse Optimal Control (IOC) or Inverse Reinforcement Learning (IRL), aim to infer the underlying objectives or preferences of agents based on observed behavior. These approaches are well-studied and have been extensively applied in single-agent settings. Typically, the objective function is assumed to be linear with respect to a set of predefined state features [13]. The goal is to estimate the weights or parameters that best align the estimated objective with the observed behavior.

Traditional IOC and IRL methods have mainly been developed for discrete state and action spaces but can also be extended to continuous settings, as seen in robotics problems. However, these works primarily focus on single-agent scenarios. In the multi-agent setting, some approaches consider cooperative or competitive agents and have been demonstrated on discretized state and action spaces [14]. More recent works have explored the multi-agent competitive setting with continuous state and action spaces, particularly in linear-quadratic games with low-dimensional states and control inputs [11].

Instead of offline estimation, where a parameter vector is identified from a set of complete trajectories, some works aim to perform online estimation as an agent within the game [11]. This means using only knowledge from the previous state to estimate the objective function and inform future actions, considering limited demonstration data and real-time execution constraints. This formulation is better

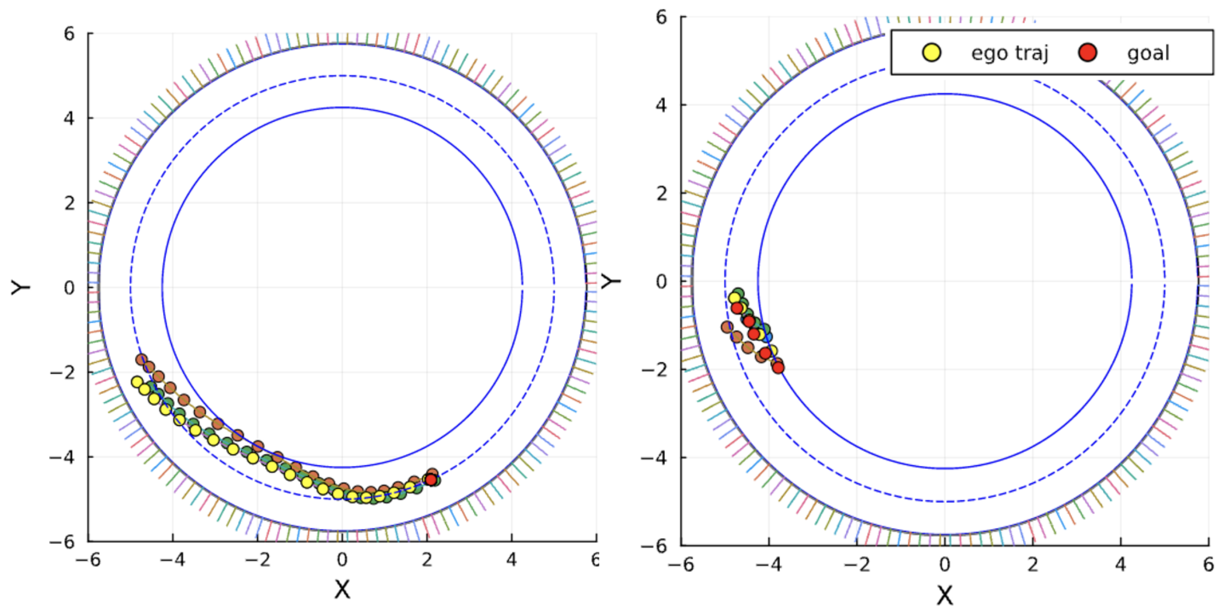

Fig. 2. We train an MLP based on long-horizon planning data (left). The output of the MLP is used for shorter-horizon planning (right). This plot (left) shows 20-step trajectories solved for during training using [2] for 3 agents. Points along the trajectory are plotted in Cartesian coordinates. This trajectory has a fixed goal state (red). During policy execution (right), the solver tracks a goal trajectory (ego goal trajectory in red) that is the output of our imitation learning policy.

suited for the online prediction-planning pipeline associated with dynamic games. However, this method is limited in that it still assumes some inherent structure of the objective function. For example, in this work, they assume a quadratic objective function with fixed goal state.

C. Learning-based Methods

In using learning-based methods, neural networks can be leveraged as function approximators for various parts of the planning pipeline. In Multi-Agent Reinforcement Learning (MARL), the policy followed by the ego agent is often represented as a neural network that is trained from data observed by interacting with opponents, as done in [15]. Other works introduce an offline learning phase in conjunction with model-based game theoretic planning methods. For example [10] warm-starts their iterative best response scheme with a learned trajectory “guess”. Alternatively [16] introduces a lifted trajectory game formulation addressing the complexity and uniqueness issues in multi-agent interactions by offloading the computational complexity to an offline training phase. By introducing auxiliary trajectory references and reference generators, the formulation reduces the runtime computation to single-agent trajectory optimization problems, which can be solved in parallel. The trained reference generators approximate solutions to the game by evaluating the references, solving trajectory optimization problems, and computing the bimatrix game. This approach enables efficient real-time decision-making in multi-agent scenarios while significantly reducing computational complexity. However, it still assumes a given objective function and structure.

In this work, we use a combination of learning-based methods and game theoretic planning to predict the objectives of other agents in a game, and plan using these predicted objectives. To do this, we train a neural network on trajectory data of all players. The neural network takes an input of the joint state of all agents in the system at the current time step and predicts the trajectories that all agents will follow in the proceeding time steps. From here, we construct objective functions for each player in the game as a tracking cost to the predicted trajectories. Finally, we use [2] as a game theoretic solver to get trajectories that satisfy a Nash equilibrium to the approximated objectives.

III. PROBLEM STATEMENT

We consider a scenario in which multiple players of a game have ground truth full state information and dynamics of themselves and all other players. We define the state vector of player ν at time step k as $\mathbf{x}_k^\nu$ and the corresponding control vector as $\mathbf{u}_k^\nu$. We further define the joint state and control of all players by removing the superscript to denote the player index:

$$\mathbf{x}_k = \left[(\mathbf{x}_k^{\nu=1})^T \quad \dots \quad (\mathbf{x}_k^{\nu=p})^T\right]^T \tag{1}$$
$$\mathbf{u}_k = \left[(\mathbf{u}_k^{\nu=1})^T \quad \dots \quad (\mathbf{u}_k^{\nu=p})^T\right]^T, \tag{2}$$

where p is the number of players in the game. The joint dynamics of the system evolve as:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k). \tag{3}$$

The objective of the ego agent is known, however the objectives of the other players are assumed to be unknown. Instead, we only have access to expert demonstrations of agents interacting in the form of trajectory data. We denote the ego agent as $\nu = 1$, and additionally define the final time step of the system's trajectory as K.

IV. APPROACH

We ground imitation learning with game-theoretic trajectory optimization. To do so, we train a policy to output an optimal trajectory for all agents in the system. Then, we use this trajectory as the goal trajectory over a fixed planning horizon. A dynamic game is then solved given these goal trajectories. Fig. 3 shows our method.

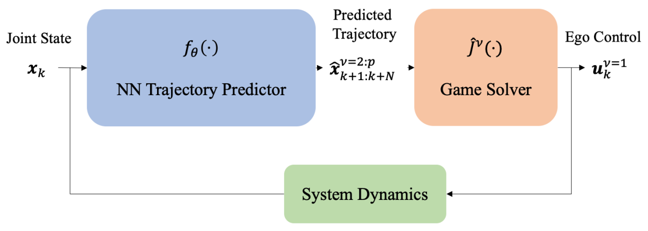

Fig. 3. We train an MLP to predict the joint trajectory of a system of agents to formulate a tracking objective that is used in a game solver.

A. Learning Objectives with Prediction Network

As described in Section III, we assume access to a dataset with expert demonstrations of agents interacting in the form of trajectory data. With this dataset, we train a neural network to map between each state to the next N states in the trajectory:

$$\hat{\mathbf{x}}^{\nu=2:p}_{k=1:N} = f_\theta(\mathbf{x}^{\nu=1:p}_{k=0}). \tag{4}$$

We use supervised learning to train this trajectory predictor with DAgger [17], rolling out the aforementioned supervised learning trajectory, then mapping each state visited to an MPC trajectory computed offline. These states are then added to the dataset and a new trajectory generator is trained.

B. Game Theoretic Planning

This predicted trajectory output of our network is then used as a tracking cost for a game solver [2]. Since we assume no access to ground truth objective functions, the following tracking cost is constructed using the neural network output with the predicted objective:

$$\underset{\mathbf{x}_{k=1:N},\mathbf{u}^\nu_{k=1:N}}{\text{minimize}} \sum_{k=1}^{N} \frac{1}{2}(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T \mathbf{Q}(\mathbf{x}_k - \hat{\mathbf{x}}_k) + \frac{1}{2}\mathbf{u}^{\nu T}_k \mathbf{R}\mathbf{u}^\nu_k \tag{5}$$

$$\text{subject to } \mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) \tag{6}$$

$$0 \geq c(\mathbf{x}_k, \mathbf{u}_k), \qquad \forall k \in \{1, \ldots, N\}, \tag{7}$$

where N is the predicted horizon length of the trajectory, $\mathbf{Q}$ is a state cost matrix, $\mathbf{R}$ is a control cost matrix, and $c(\cdot)$ is a collision avoidance cost both between agents and between agents and the environment. With this predicted objective, our approach can be used in scenarios where agents have underlying ground truth objectives of varying structure.

V. RESULTS

A. Simulated Data Generation

We consider a setting where three players interact on a circular track, and plan toward a single goal state that recedes around the track as a function of the ego agent's current state. We use a state vector of position and velocity, and a control vector of acceleration:

$$\mathbf{x}^\nu_k = \begin{bmatrix}\mathbf{p}^{\nu T}_k & \mathbf{v}^{\nu T}_k\end{bmatrix}^T = \begin{bmatrix}x & y & z & v_x & v_y & v_z\end{bmatrix}^T \tag{8}$$

$$\mathbf{u}^\nu_k = \begin{bmatrix}a_x & a_y & a_z\end{bmatrix}^T, \tag{9}$$

and use double integrator dynamics to propagate the states:

$$\mathbf{p}^\nu_{k+1} = \mathbf{p}^\nu_k + \mathbf{v}^\nu_k \Delta t \tag{10}$$

$$\mathbf{v}^\nu_{k+1} = \mathbf{v}^\nu_k + \mathbf{u}^\nu_k \Delta t, \tag{11}$$

where Δt is the time step in seconds.

We generate a dataset using [2], where all three players have the following ground truth objective:

$$J^\nu(\mathbf{x}_{k=1:N}, \mathbf{u}^\nu_{k=1:N}) = \sum_{k=1}^{H} \frac{1}{2}(\mathbf{x}_k - \mathbf{x}_g)^T \mathbf{Q}(\mathbf{x}_k - \mathbf{x}_g) + \frac{1}{2}\mathbf{u}^{\nu T}_k \mathbf{R}\mathbf{u}^\nu_k, \tag{12}$$

where H is the planning horizon used by the ground truth objective function during data collection, and $\mathbf{x}_g$ is a goal state that recedes around the track as a function of the ego agent's state. For the training dataset, we use $H = 20$. Full state information is stored for all three players over 1000 K-step MPC rollouts. Our dataset maps ground truth joint states, to the next N joint states for the multi-agent system, defining a predictive horizon of $N = 5$.

B. Short Horizon Comparison

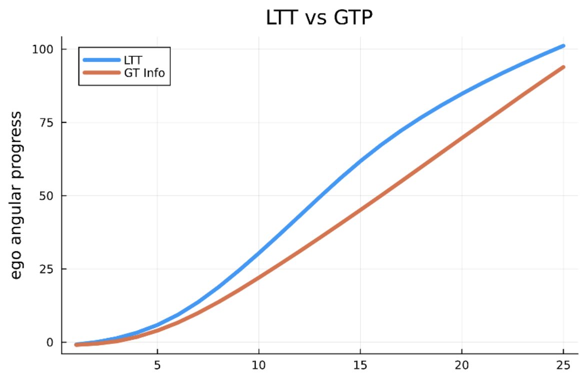

Fig. 4. LTT agents make more angular progress than GTP agents with ground truth objective information. This shows that our offline training phase with DAgger effectively imbues longer-horizon planning information into the short-horizon planning during policy execution.

We compare performance of an agent using our method, which we denote LTT for Learned Trajectory Tracking, to an agent planning with standard Game Theoretic Planning (GTP) as described in [2]. LTT is trained from trajectories with a horizon $H = 20$ using Model Predictive Control

(MPC). We first test LTT with an $N = 5$ predictive planning horizon and compare to standard GTP with an $H = 5$ planning horizon during policy execution. We show that LTT (which does not have ground truth objective function information) on average progresses 101.15 degrees along the circular track based on 1000 rollouts of MPC trajectories of length $K = 25$, as compared to only 93.88 degrees for standard GTP, which has ground truth objective information. We believe that LTT is able to outperform GTP because the long-horizon planning it is trained off of imbues longer-horizon information in the trajectories it tracks. This comparison is shown in Fig. 4.

C. *Long Horizon Comparison*

Next, LTT is compared to GTP with varying planning horizons during execution. LTT performs most similarly to standard GTP with planning horizon of $H = 20$, when LTT has planning horizon of $N = 5$ online. This makes sense, because LTT was trained from trajectories planned with $H = 20$. This is an important result because the GTP agents have access to the ground truth objectives of the players while the LTT agents must approximate the objectives. This result is shown in Fig. 5.

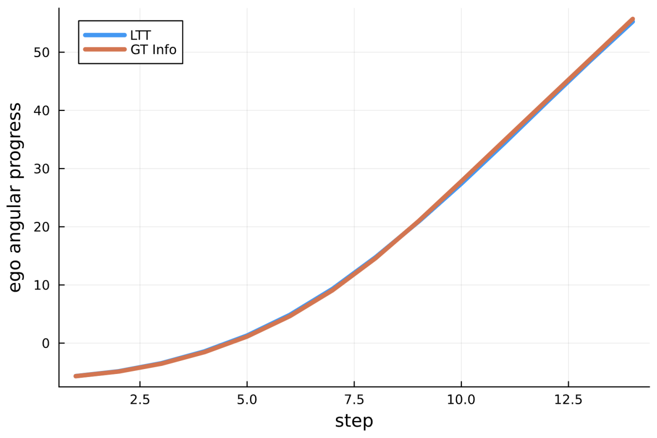

Fig. 5. LTT with N=5 online MPC has similar performance to standard GTP with H=20. Angular progress along the track is plotted as a function of MPC step.

Finally, we tested the runtime complexity of our method. On an M2 Mac, an $H = 20$ step MPC plan takes around 0.13 seconds to compute, whereas a $H = 5$ step MPC plan takes about 0.025 seconds to compute. As these methods have similar performance we find that LTT decreases runtime by 80.9 percent.

VI. CONCLUSIONS

We present a formalized game theoretic framework to ground neural network generated trajectories with trajectory optimization. We show that learning from agents using long-horizon planning can allow us to decrease online planning horizons for agents executing game theoretic planning online. Furthermore we show that using a prediction network that outputs multi-agent trajectories can be used in place of a fixed "goal state" for objective of agents. This offline training phase is effective at imbuing long-horizon planning data into a short-horizon online planning execution.

A feature of our method is that it can be combined with other trajectory prediction frameworks, for which their is a robust body of literature. While we use an offline training phase with DAgger, we can use any architecture for trajectory prediction, some of which may be better suited for different problems.

In future work, we plan to test our method in cases where the underlying ground truth objective does not have a quadratic structure. We also plan to compare our method to other objective-function estimation methods.

REFERENCES

[1] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 9590–9596.
[2] S. L. Cleac'h, M. Schwager, and Z. Manchester, "Algames: A fast solver for constrained dynamic games," *arXiv preprint arXiv:1910.09713*, 2019.
[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.
[4] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.
[5] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 66–73.
[6] A. Liniger and J. Lygeros, "A noncooperative game approach to autonomous racing," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 884–897, 2019.
[7] G. Williams, B. Goldfain, P. Drews, J. M. Rehg, and E. A. Theodorou, "Best response model predictive control for agile interactions between autonomous ground vehicles," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2403–2410.
[8] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game theoretic planning for self-driving cars in competitive scenarios." in *Robotics: Science and Systems*, 2019, pp. 1–9.
[9] R. Spica, E. Cristofalo, Z. Wang, E. Montijano, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1389–1403, 2020.
[10] Z. Wang, T. Taubner, and M. Schwager, "Multi-agent sensitivity enhanced iterative best response: A real-time game theoretic planner for drone racing in 3d environments," *Robotics and Autonomous Systems*, vol. 125, p. 103410, 2020.
[11] S. Le Cleac'h, M. Schwager, and Z. Manchester, "Lucidgames: Online unscented inverse dynamic games for adaptive trajectory prediction and planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5485–5492, 2021.
[12] 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," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 1475–1481.
[13] N. D. Ratliff, "Learning to search: Structured prediction techniques for imitation learning," Ph.D. dissertation, Carnegie Mellon University, 2009.
[14] 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, 2010.
[15] M. Shen and J. P. How, "Safe adaptation in multiagent competition," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 12 441–12 447.

[16] L. Peters, D. Fridovich-Keil, V. Rubies-Royo, C. J. Tomlin, and C. Stachniss, “Inferring objectives in continuous dynamic games from noise-corrupted partial state observations,” *arXiv preprint arXiv:2106.03611*, 2021.

[17] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.