Game-Theoretic Planning for Self-Driving Cars in Multivehicle Competitive Scenarios

Mingyu Wang, Zijian Wang, John Talbot, J. Christian Gerdes, *Member, IEEE*, and Mac Schwager, *Member, IEEE*

***Abstract*—In this article, we propose a nonlinear receding horizon game-theoretic planner for autonomous cars in competitive scenarios with other cars. The online planner is specifically formulated for a multiple-car autonomous racing game, in which each car tries to advance along a given track as far as possible with respect to the other cars. The algorithm extends previous work on game-theoretic planning for single-integrator agents to be suitable for autonomous cars in the following ways: 1) by representing the trajectory as a piecewise polynomial; 2) incorporating bicycle kinematics into the trajectory; and 3) enforcing constraints on path curvature and acceleration. The game-theoretic planner iteratively plans a trajectory for the ego vehicle and then the other vehicles in sequence until convergence. Crucially, the trajectory optimization includes a sensitivity term that allows the ego vehicle to reason about how much the other vehicles will yield to the ego vehicle to avoid collisions. The resulting trajectories for the ego vehicle exhibit rich game strategies such as blocking, faking, and opportunistic overtaking. The game-theoretic planner is shown to significantly outperform a baseline planner using model-predictive control, which does not take interaction into account. The performance is validated in high-fidelity numerical simulations with three cars, in experiments with two small-scale autonomous cars, and in experiments with a full-scale autonomous car racing against a simulated vehicle (video is available at https://youtube.com/playlist?list=PLmIcLEh8KMje4rYBqRANDuKvqFvj7LCRp).**

***Index Terms*—Motion planning, multirobot systems.**

I. Introduction

In this article, we seek to enable an autonomous car to interact with other vehicles, both human-driven and autonomous, while reasoning about other vehicles' intentions and responses. Interactions among multiple cars are complicated because there is no centralized coordination, and there exist both elements of cooperation and competition during the process. On the one hand, all agents must cooperate to avoid collisions. On the other hand, competition is often present in autonomous driving scenarios, such as merging, lane changing, overtaking, and so on. In these situations, the action of one agent is dependent on the intentions of other agents, and *vice versa*, thus exhibiting rich dynamic behaviors that are seldom characterized in the existing literature.

We propose a game-theoretic planning algorithm that explicitly handles such complex phenomena. In particular, we exploit the concept of Nash equilibria from game theory, which are characterized by a set of actions such that no single agent can do better by changing its action unilaterally. We develop our algorithm for a multiple-car racing game, in which the objective of each car is to advance along the track as far beyond its competitors as possible. The vehicles also share common collision avoidance constraints to formalize the intention that all cars want to avoid collisions. Nevertheless, there is no means of explicit communication between the vehicles. It is through the collision avoidance constraints that one agent can impose influence on the other agents and also can acquire information by observing or probing the opponent. We use an iterative algorithm, called sensitivity-enhanced iterated best response (SE-IBR), that seeks to converge to a Nash equilibrium in the joint trajectory space of all the agents. The ego vehicle iteratively solves several rounds of trajectory optimization, first for itself, then for all opponents, then again for itself, etc. The fixed point of this iteration represents a Nash equilibrium in the trajectory space of all vehicles; hence, the trajectory for the ego vehicle accounts for the predicted response of the other vehicles. The planned trajectories are specifically suited to cars as we parameterize them by piecewise time polynomials, which take into account the nonholonomic vehicle kinematics and also enforce bounded steering and acceleration constraints.

In order to demonstrate the effectiveness of our model, we verify our algorithm in both simulations and experiments. The simulations use a high-fidelity vehicle dynamics model and show that our game-theoretic planner (GTP) significantly outperforms opponents using conservative model-predictive control (MPC)-based trajectory planners, which only avoid collisions passively. We also perform experiments with small-scale model cars and a full-sized autonomous car. For the model cars, we show again that the car using our GTP significantly outruns its competitor with the aforementioned MPC-based planner. A snapshot from an experimental run is shown in Fig. 1. For the full-scale car (shown in Fig. 2), we show that the GTP outperforms a

Manuscript received March 9, 2020; accepted November 24, 2020. This article was recommended for publication by Associate Editor J. O'Kane and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. This work was supported by Toyota Research Institute (TRI). This article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. *(Corresponding author: Mingyu Wang.)*

Mingyu Wang, John Talbot, and J. Christian Gerdes are with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: mingyuw@stanford.edu; john.talbot@stanford.edu; gerdes@stanford.edu).

Zijian Wang and Mac Schwager are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA (e-mail: zjwang@alumni.stanford.edu; schwager@stanford.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2021.3047521.

Digital Object Identifier 10.1109/TRO.2020.3047521

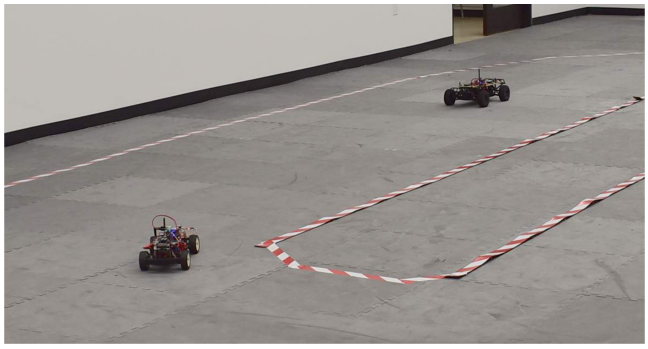


Fig. 1. Indoor experiments using RC car experimental platform. The motion capture system is used for odometry information of both cars.

Fig. 2. Outdoor experiments using X1—a drive-by-wire research vehicle at Thunderhill Raceway Park, CA, USA.

simulated competing vehicle (simulated for safety reasons) using the aforementioned MPC-based planner.

A preliminary version of some of the work in this article appeared in the conference paper [1], which discusses two-car competitive zero-sum games. In this article, we build upon the previous work to handle multiple-car competitive scenarios, which are not necessarily zero-sum games. Real-time simulations with three cars are provided, and results are discussed. We also provide more comprehensive technical derivations.

The organization of this article is as follows. Section II gives an overview of the related works. Section III describes the car model and the formulation of multiple-car competitive games. The concepts of Nash equilibria and SE-IBR algorithm are also given. The incorporation of car dynamics into the game-theoretic planning approach is discussed in Section IV. Simulations and experiments with analysis of the resulting performance are described in Sections V and VI, respectively. Finally, Section VII concludes this article.

II. Related Work

Our work builds upon the results in [2], in which the SE-IBR algorithm was proposed for drone racing, assuming single-integrator dynamics for the drones and representing the planned trajectories as discrete waypoints. Recently, in [3] and [4], Wang *et al.* have extended this algorithm to multiple drones (more than two) in 3-D environments, again with single-integrator dynamics. The main innovation in our work beyond these is: 1) to represent the trajectory as a continuous-time piecewise polynomial; 2) to incorporate bicycle kinematic constraints; and 3) to include turning angle and acceleration constraints. All of these advancements are for the purpose of making this algorithm suitable for car racing. We also provide new simulation results with a high-fidelity car dynamics simulator, experimental results with small-scale autonomous race cars in the laboratory, and experimental results with a full-scale autonomous car on a test track.

Some other recent works have proposed iterated best response (IBR) algorithms to reach Nash equilibria in multivehicle interaction scenarios. Williams *et al.* [5] propose an IBR algorithm together with an information-theoretic planner for controlling two ground vehicles in close proximity. In that work, the vehicles are not competing against each other, but rather collaborating to maintain a relative position while competing against the effects of noise. In contrast, in our problem, the cars compete against one another, and the only coupling between the two vehicles is through a collision avoidance constraint, which is handled by our SE-IBR algorithm by adding a sensitivity term to the best response iterations. Furthermore, the algorithm in [5] uses an information-theoretic planner in each iteration, which accommodates stochasticity, while ours uses a nonlinear MPC approach with continuous-time vehicle trajectories, which allows us to incorporate realistic kinematic constraints. In a similar vein, Sadigh *et al.* [6] propose a game-theoretic approach for an autonomous car to interact with a human driven car. That work proposes a Stackelberg solution (as opposed to a Nash solution) with potential field models to account for human driver behavior. It also does not consider constraints (e.g., road or collision constraints) for the vehicles. Fisac *et al.* [7] decompose the trajectory planning task during interaction into a strategic level and a tactical level to enable real-time performance. On the strategic level, dynamic programming is used to solve a dynamic game. Thornton *et al.* [8] propose a speed control algorithm for cars near occluded pedestrian crosswalks, where value sensitive design methodology is applied to a partially observable Markov decision process framework. A thorough overview of motion planning for autonomous vehicles is provided in [9] and [10].

Aside from the above examples, the existing literature in game-theoretic motion planning and modeling for multiple competing agents typically is either for discrete state or action spaces [11]–[13] or is specifically for pursuit-evasion style games. In [14], a hierarchical reasoning game theory approach is used for interactive driver modeling; Bahram *et al.* [15] predict the motion of vehicles in a model-based intention-aware framework using an extensive-form game formulation; Liniger *et al.* [16], [17] consider Nash and Stackelberg equilibria in a two-car racing game, where the action space of both players is finite and the game is formulated in bimatrix form. A similar approach is also used to model human collision avoidance behavior in an extensive-form game [18]. In contrast to these previous works, we consider a continuous trajectory space, which can capture the

complex interactions and realistic dynamics of cars. Differential games are studied in [19]–[22] in the context of a pursuit-evasion game, where pursuers and evaders are antagonistic toward each other. This antagonistic assumption is unrealistic in real-world driving and leads to over conservative behaviors for the ego vehicle. More recently, in [23], an iterative linear–quadratic game approach is used to solve for a Nash equilibrium of nonlinear games. Generalized Nash equilibrium problems are also of great interests to economics field. In [24], properties of generalized Nash equilibria in two-player games are discussed. Furthermore, for linear–quadratic-type games, an equilibrium selection method is introduced in [25].

There is also a vast body of work on collision avoidance in a non-game-theoretic framework. To name just a few, the authors of [26] and [27] present a provable distributed collision avoidance framework for multiple robots using Voronoi diagrams, under the assumption that agents act reciprocally. Control barrier functions are used in [28] for adaptive cruise control. The paper [29] presents a control structure for emergency scenarios, where cars have to maneuver at their limits to achieve stability and avoid collisions with static obstacles. The paper [30] introduces a minimally interventional controller for closed-loop collision avoidance using backward reachability analysis. By contrast, we focus on motion planning in a competitive multicar racing game, where agents are adversarial and have influence on each other through shared collision avoidance constraints.

III. Problem Statement

In this section, we first describe the kinematic bicycle model used to model cars and the necessary notations to formulate the multiple-car racing game, as given in [1]. Following this, we concisely summarize the GTP introduced in [2] and [3], which was proposed for single-integrator robots and has been successfully applied for drone racing. However, previous work does not consider robots with nonlinear or nonholonomic dynamics, such as cars.

A. Bicycle Model of Cars

To plan a feasible trajectory for cars, our planner requires a more realistic motion model than a single or double integrator. At the same time, we want to balance model fidelity and computation complexity. For this purpose, we use the following kinematic bicycle model, which consists of four states, as illustrated in Fig. 3:

$$\begin{aligned}\dot{x}(t) &= v(t)\cos\theta(t)\\ \dot{y}(t) &= v(t)\sin\theta(t)\\ \dot{\theta}(t) &= \frac{v(t)}{L}\tan\phi(t)\\ \dot{v}(t) &= a(t)\end{aligned} \tag{1}$$

where x, y, v, and θ are 2-D positions, speed, and heading of the car, respectively, a and ϕ are acceleration and steering angle commands, respectively, and L is the distance between front and rear axles. The bicycle model incorporates the nonholonomic steering constraint present in real cars and is suitable for real-time planning.

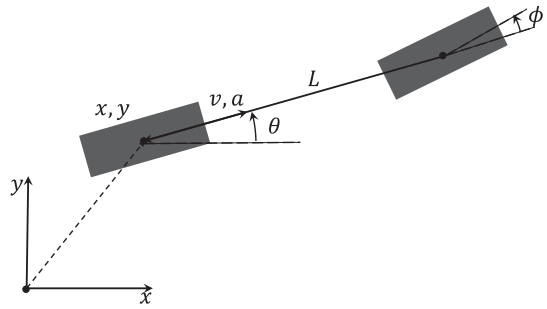


Fig. 3. Kinematic bicycle model with four states: position (x, y), speed (v), and heading (θ).

We note that the above bicycle model can be shown to be differentially flat [31], [32] with the flat output $\boldsymbol{\sigma} = [\sigma_1,\ \sigma_2]^T = [x,\ y]^T$. Briefly, a dynamic system $\dot{\mathbf{x}} = f(\mathbf{x}, \boldsymbol{u})$ with state vector $\mathbf{x}$ and control vector $\boldsymbol{u}$ is differentially flat if there exists a flat output vector σ and smooth functions α, h, and ϕ such that

$$\sigma = h(\mathbf{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, \ldots, \boldsymbol{u}^{(r)})$$

and the state and control input of the dynamic system can be written as a function of σ and its time derivatives

$$\begin{aligned}\mathbf{x} &= \phi(\sigma, \dot{\sigma}, \ldots, \sigma^{(p)})\\ \boldsymbol{u} &= \alpha(\sigma, \dot{\sigma}, \ldots, \sigma^{(p)}).\end{aligned}$$

For this reason, we can analytically compute the control inputs and all state variables for the car given trajectories of the flat outputs and their derivatives up to some order p. In the case of bicycle models, $p = 2$. The functions $\phi(\cdot)$ and $\alpha(\cdot)$ are called endogenous transformation. For the kinematic bicycle model, these functions are given as

$$\begin{aligned}x &= \sigma_1\\ y &= \sigma_2\\ \theta &= \arctan 2(\dot{\sigma_2}, \dot{\sigma_1})\\ v &= \sqrt{\dot{\sigma_1}^2 + \dot{\sigma_2}^2}\\ a &= \frac{\dot{\sigma}_1\ddot{\sigma}_1 + \dot{\sigma}_2\ddot{\sigma}_2}{\sqrt{\dot{\sigma}_1^2 + \dot{\sigma}_2^2}}\\ \phi &= \arctan\left(\frac{\dot{\sigma}_1\ddot{\sigma}_2 - \dot{\sigma}_2\ddot{\sigma}_1}{(\dot{\sigma}_1^2 + \dot{\sigma}_2^2)^{\frac{3}{2}}}L\right).\end{aligned} \tag{2}$$

To exploit this property, we use the flat output $\boldsymbol{\sigma} = [\sigma_1,\ \sigma_2]^T$ and plan trajectories in flat output space instead of the original state space. Each flat output trajectory is represented by a second-order piecewise polynomial. By doing so, we convert the nonlinear dynamics constraints into piecewise polynomial trajectory representations. Furthermore, open-loop control inputs can be derived from the piecewise polynomials afterward using (2). The endogenous transformations also enable us encode control input constraints in terms of trajectory parameters.

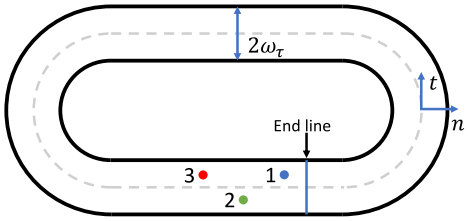


Fig. 4. Track diagram for multiple-car competitive games. Gray-dashed line is the center line of the track. Black lines are the track boundaries. Colored dots are the starting positions for three players, respectively.

B. Multiple-Car Racing Game

We now define the multiple-car competing game. Consider R cars competing with each other on a predefined known track. We discretize time and let $\mathbf{p}_i^n \in \mathbb{R}^2$ and $\mathbf{u}_i^n \in \mathbb{R}^2$ be robot i's position and velocity at time step n. The dynamics model for each player follows the continuous representation in (1). The racing track is characterized by its center line

$$\boldsymbol{\tau} : [0, l_{\boldsymbol{\tau}}] \mapsto \mathbb{R}^2$$

where $l_{\boldsymbol{\tau}}$ is the total track length. Track tangent and normal vectors are denoted as

$$\boldsymbol{t} = \boldsymbol{\tau}',$$
$$\boldsymbol{n} = \boldsymbol{\tau}''.$$

Track width is $w_{\boldsymbol{\tau}}$. A diagram of an example track is shown in Fig. 4. These track parameters are known by all racing agents beforehand. The progress of player i along the track is defined as the arc length s of the closest point to the robot's current position $\mathbf{p}_i$ on the track center line

$$s_i(\mathbf{p}_i) = \arg\min_s \frac{1}{2}||\boldsymbol{\tau}(s) - \mathbf{p}_i||^2. \tag{3}$$

The objective of a player is to travel along the track as far as possible, which can be formulated by the following optimization problem:

$$\max_{\boldsymbol{\theta}_i \in \Theta_i} \; s_i(\mathbf{p}_i^N) \tag{4}$$

where $\boldsymbol{\theta}_i$ is the decision variable vector, including the trajectory parameters for player i. Here, Θ_i is the feasible space, which takes into account all constraints. There is a slight overloading of notations here. We use vector representation $\boldsymbol{\theta}_i$ to denote decision variables for player i, while scalar θ represents vehicle heading in Section III-A. We adopt a receding horizon planning approach with planning horizon of N steps. Only the first step of the planned trajectory will be executed, and the optimization problem will be solved again at the next time step. The objective function in (4) represents the motivation of one player to travel further at the end of planning horizon. All players' objective functions are available to every other player, which is reasonable in the case of a racing game, since all players have the same objective of getting as far along the track as quickly as possible.

Constraints of the optimization problem include dynamics and control input constraints, collision avoidance constraints, and track constraints. The dynamics and control input constraints will be explained in detail in Section IV. The collision avoidance constraint for player i is

$$\left\|\mathbf{p}_j^n - \mathbf{p}_i^n\right\| \geq \overline{d}_i \quad \forall n, \forall j \neq i \tag{5}$$

where $\overline{d}_i$ is the minimum clearance distance for agent i to avoid collisions (which may be different or the same for different agents and can be used to represent different risk tolerances for collisions). Theoretically, we could use more complex model to impose collision avoidance constraints, for example, with ellipsoids. However, circular constraints are used here for computational tractability and real-time planning requirements.

Track constraints keep the player inside the track boundaries

$$\left|\boldsymbol{n}(\mathbf{p}_i^n)^T[\mathbf{p}_i^n - \boldsymbol{\tau}(\mathbf{p}_i^n)]\right| \leq w_{\boldsymbol{\tau}} \quad \forall n. \tag{6}$$

Here, we express the track normal vector $\boldsymbol{n}$ to be a function of a vehicles position, representing the normal vector of the point on track center line closest to the vehicle. Similarly, $\boldsymbol{\tau} : \mathbb{R}^2 \to \mathbb{R}^2$ is a function from a vehicle's position to the closest point on the track center line.

In this optimization problem (4), player i only has control authority over its own planned trajectory $\boldsymbol{\theta}_i$. Other players' planned trajectories $\boldsymbol{\theta}_j, j \neq i$, are not available to player i. Due to the competitive nature of this game, it is also unrealistic to assume that we can rely on opponents to communicate their strategies. However, because of the collision avoidance constraints (5), player i needs $\boldsymbol{\theta}_j$s to calculate its optimal solution. In other words, the feasible trajectory set of player i, Θ_i, depends on the unknown trajectories of all opponents $j \neq i$.

C. Nash Equilibria in Racing Games

To capture the game nature of this problem, we use the concept of Nash equilibria [33]. Unlike in [6], where the robot car is assumed to take an action first and the human driven car responds (which leads to a Stackelberg game), in our work, no player has information advantage over its opponents. Thus, Nash equilibria better suit the problem. A Nash equilibrium is a strategy profile (a set of trajectory plans, in our case) where no player can do better by unilaterally changing its strategy. Formally, $\Theta = \Theta_1 \times \Theta_2 \cdots \Theta_R$ is the set of feasible strategy tuples for all R players, and Θ_i is the set of feasible strategies of player i given the strategy of other players, $f_i(\boldsymbol{\theta})$ is the payoff function for player i evaluated at $\boldsymbol{\theta} \in \Theta$. $\boldsymbol{\theta}_i \in \Theta_i$ denotes the strategy profile of player i, and $\boldsymbol{\theta}_{-i}$ denotes the strategy profile of all players other than i. We use $(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i})$ to highlight that player i only has direct control authority over $\boldsymbol{\theta}_i$, while $\boldsymbol{\theta}_{-i}$ represent the decision variables of other players. This is only for notation simplicity and does not affect the meaning of $\boldsymbol{\theta}$.

A strategy profile $\boldsymbol{\theta}^* = (\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^*, \cdots \boldsymbol{\theta}_R^*)$ is a Nash equilibrium if

$$\boldsymbol{\theta}_i^* = \arg\max_{\boldsymbol{\theta}_i \in \Theta_i(\boldsymbol{\theta}_{-i}^*)} f_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}^*), \quad i = 1, \ldots, R. \tag{7}$$

Computing Nash equilibria, in general, can be computationally intractable [34], and there can be multiple or an infinite number of Nash equilibria in a game [24]. Thus, we resort to an iterative algorithm to asymptotically approach a Nash equilibrium if one exists, as will be explained in Section III-D. If our algorithm converges, it is proven to converge to a Nash equilibrium. The algorithm is not mathematically guaranteed to converge, but, empirically, we find that it does so except in very rare cases. In these cases, the receding horizon nature of our algorithm ensures that the trajectory will be immediately replanned and a new Nash equilibrium can be found.

We note that in our problem formulation, the payoff function f_i only depends on ego player's strategy $\boldsymbol{\theta}_i$ and, in general, is independent of other players strategy $\boldsymbol{\theta}_{-i}$. Each player's objective value is only a function of its own final position. However, as we can observe from (7), $\boldsymbol{\theta}_{-i}$ indeed has influence on $\boldsymbol{\theta}_i^*$ at a Nash equilibrium through the feasible set constraints, i.e., the optimal value at a Nash equilibrium *is* a function of $\boldsymbol{\theta}_{-i}$. We will leverage this property to enhance the IBR method to approximate a Nash equilibrium.

We also note that in our game formulation, we require that all players know the optimization problems all other players are solving. While this is a strong assumption in general traffic settings, it is more realistic in competitive games where the objective function of each player is to advance along a given track.

D. Iterated Best Response and Sensitivity-Enhanced Objective Function

To simplify the notation, (4) can be rewritten as

$$\max_{\boldsymbol{\theta}_i} \; s_i(\boldsymbol{\theta}_i) \tag{8a}$$

$$\text{s.t.} \quad h_i(\boldsymbol{\theta}_i) = 0 \tag{8b}$$

$$g_i(\boldsymbol{\theta}_i) \leq 0 \tag{8c}$$

$$\gamma_{ij}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \leq 0 \quad \forall j \neq i \tag{8d}$$

where $h_i(\cdot)$ represents a vector of equality constraints, $g_i(\cdot)$ represents a vector of inequality constraints, and $\gamma_{ij}(\cdot,\cdot)$ is a vector of inequality constraints, which couple the ego player's decision with its opponents' decisions. As we pointed out, although each player's objective is solely a function of its own decision and other players do not have control authority of the strategy of player i, they can still influence the ego player's optimal strategy through the coupled collision avoidance constraint (5). In other words, other players' strategies $\boldsymbol{\theta}_{-i}$ do affect the payoff of player i at a Nash equilibrium. Thus, the optimal payoff of player i can be represented as a function of $\boldsymbol{\theta}_{-i}$, i.e., $s_i^*(\boldsymbol{\theta}_{-i})$. Similarly, the ego player's strategy also affects the other players' optimal strategies at a Nash equilibrium. To incorporate the influence of one player on its opponents' optimal payoff, we propose to substitute the objective in problem (8) by

$$s_i(\boldsymbol{\theta}_i) - \sum_{j \neq i} \alpha_{ij} s_j^*(\boldsymbol{\theta}_i) \tag{9}$$

which is a combination of the reward of a player itself and negative rewards of all other players. The second term is motivated by the competitive nature of this game. In addition to advancing along the track, each player is also motivated to decrease the other players' rewards by changing its own strategy. Crucially, we prove that a Nash equilibrium with this new objective function is also a Nash equilibrium of our original game. Hence, solving the game with the surrogate objective function still leads to a solution for the original game.

A closed-form solution of $s_j^*(\boldsymbol{\theta}_i)$ is not easily available. However, we can apply sensitivity analysis [35] and get a linear approximation of $s_j^*(\boldsymbol{\theta}_i)$ around an available solution of $\boldsymbol{\theta}_i$. An SE-IBR algorithm is used in our algorithm. The ego vehicle starts with an initial guess of all opponents' trajectories and solves the optimization problem (8) with sensitivity-enhanced objective (9) for the ego player. The ego player then solves iteratively for all other players until a convergence criterion is met. At each iteration, the ego vehicle sequentially solves (8) with objective (9) for all opponents. Each iteration ends with the ego player. When solving the problem for each player, the ego vehicle uses the solutions of all other players' strategies from the previous iteration.

Specifically, assume that in the lth iteration, the optimal solutions for all players from the last iteration are $\boldsymbol{\theta}^l$. Then, when player i solves the optimization for player j to obtain its opponents' optimal value $s_j^*(\boldsymbol{\theta}_i^l)$, $\boldsymbol{\theta}_i^l$ and $\boldsymbol{\theta}_j^l$ are treated as fixed. If we linearize $s_j^*(\boldsymbol{\theta}_i)$ around $\boldsymbol{\theta}_i^l$, then

$$s_j^*(\boldsymbol{\theta}_i) = s_j^*(\boldsymbol{\theta}_i^l) + \left.\frac{ds_j^*}{d\boldsymbol{\theta}_i}\right|_{\boldsymbol{\theta}_i = \boldsymbol{\theta}_i^l} (\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^l). \tag{10}$$

It is derived in [2] that

$$\left.\frac{ds_j^*}{d\boldsymbol{\theta}_i}\right|_{\boldsymbol{\theta}_i^l} = -\boldsymbol{\mu}_{ji}^l \left.\frac{\partial \gamma_{ji}}{\partial \boldsymbol{\theta}_i}\right|_{(\boldsymbol{\theta}_i^l, \boldsymbol{\theta}_j^l)} \tag{11}$$

where $\boldsymbol{\mu}_{ji}^l$ is a row vector representing the Lagrange multiplier associated with the inequality constraints γ_{ji}. Combining (9)–(11) and neglecting the constant terms, the optimization problem that each player should solve is

$$\max_{\boldsymbol{\theta}_i \in \Theta_i^l} s_i(\boldsymbol{\theta}_i) + \sum_{j \neq i} \alpha_{ij} \boldsymbol{\mu}_j^l \left.\frac{\partial \gamma_j}{\partial \boldsymbol{\theta}_i}\right|_{(\boldsymbol{\theta}_i^{l-1}, \boldsymbol{\theta}_j^l)} \boldsymbol{\theta}_i.$$

Note that the computation of $s_i(\boldsymbol{\theta}_i)$ also requires an optimization problem as given in (3). Again, a linear approximation of this term by sensitivity analysis is obtained. The final explicit form of the optimization problem is as follows:

$$\max_{\boldsymbol{\theta}_i \in \Theta_i^l} \boldsymbol{t}^T \mathbf{p}_i^N + \sum_{j \neq i} \alpha_{ij} \boldsymbol{\mu}_j^l \left.\frac{\partial \gamma_j}{\partial \boldsymbol{\theta}_i}\right|_{(\boldsymbol{\theta}_i^{l-1}, \boldsymbol{\theta}_j^l)} \boldsymbol{\theta}_i. \tag{12}$$

Using this sensitivity-enhanced objective function, it is proved in [2] and [4] that if the SE-IBR iteration converges, then the resulting strategy tuple satisfies the necessary conditions for a Nash equilibrium. For a detailed derivation of the results, the readers are referred to [2] and [4].

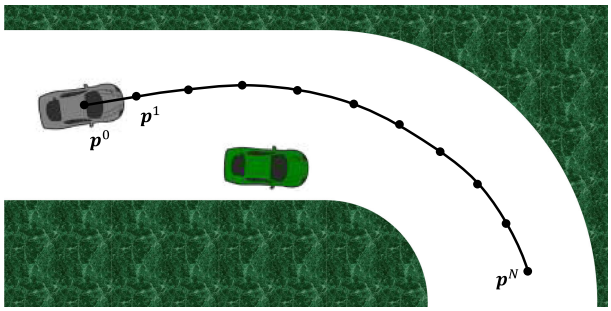


Fig. 5. Piecewise polynomial trajectory over an N-step planning horizon. Position of the nth waypoint is $\mathbf{p}^n$. Each trajectory section between two consecutive waypoints is represented using a second-order time polynomial.

IV. Car Game-Theoretic Planner

The goal of this work is to extend the previously proposed GTP to handle car dynamics to accommodate more realistic scenarios. First, the original game planner uses single-integrator dynamics for players. While this model works well with quadrotors, it is unrealistic for cars with nonholonomic constraints and limited acceleration and turning radius. We deal with a bicycle model with acceleration and steering constraints in this article. Second, we leverage the fact that the bicycle model is differentially flat and use a piecewise polynomial trajectory representation in the flat outputs. The previous formulation is modified so that an admissible solution to our optimization problem reflects the actuation limitations of cars.

A. Piecewise Polynomial Trajectory Representation

We use piecewise time polynomial trajectories to represent the planned flat outputs, i.e., $x(t)$ and $y(t)$. The advantage of using polynomials is that we can explicitly impose the constraints on vehicle states as functions of polynomial coefficients.

In 2-D space, let N be the number of polynomials for each dimension. Each polynomial has a fixed time length Δt. The starting time of the nth polynomial is denoted $t_n = n\Delta t$. The nth polynomials in two dimensions are given as follows:

$$\begin{aligned} x_n(t) &= \sum_{p=0}^{2} A_{n,p} t^p \\ y_n(t) &= \sum_{p=0}^{2} B_{n,p} t^p, \quad t \in [t_n, t_{n+1}] \end{aligned} \tag{13}$$

for $n = 0, 1, \ldots, N-1$, where $\boldsymbol{A}_n = [A_{n,0}, A_{n,1}, A_{n,2}]$ and $\boldsymbol{B}_n = [B_{n,0}, B_{n,1}, B_{n,2}]$ are the coefficients for the nth polynomial in x and y directions, respectively. Similar to the previous formulation, we also define $N+1$ waypoints with positions $\mathbf{p}^n \in \mathbb{R}^2$ and velocities $\boldsymbol{u}^n \in \mathbb{R}^2$, $n = 0, 1, 2, \ldots, N$. An illustration of the waypoints and piecewise polynomial trajectories is given in Fig. 5.

The following constraints are imposed regarding trajectory continuity and control inputs.

1) *Continuity constraints:* We enforce the position and velocity to be continuous at the waypoints

$$\begin{aligned} [x_n(t_n), y_n(t_n)] &= \mathbf{p}^n \\ [x_n(t_{n+1}), y_n(t_{n+1})] &= \mathbf{p}^{n+1} \\ [\dot{x}_n(t_n), \dot{y}_n(t_n)] &= \boldsymbol{u}^n \\ [\dot{x}_n(t_{n+1}), \dot{y}_n(t_{n+1})] &= \boldsymbol{u}^{n+1} \end{aligned} \tag{14}$$

for $n = 0, 1, \ldots, N-1$. Substituting (13) into the above equations, we obtain a set of equality constraints in $\boldsymbol{A}_n$ and $\boldsymbol{B}_n$.
We also enforce the following constraints on speed, acceleration, and curvature for each piece of the polynomial. For clarity of notation, we omit subscript n for the nth polynomial, unless there is possible confusion.

2) *Speed constraints:* From (2), we have

$$v^2(t) = \dot{x}^2(t) + \dot{y}^2(t)$$

which is a second-order polynomial in t. Substituting (13), we obtain coefficients, which are functions of $A_{\cdot,\cdot}$s and $B_{\cdot,\cdot}$s. Notice that the maximum value of speed in the interval $t \in [t_n, t_{n+1}]$ is attained at its end points. Thus, we have $v_{n,\max}(t) = \max\{||\boldsymbol{u}^n||, ||\boldsymbol{u}^{n+1}||\}$. Velocity constraints are

$$||\boldsymbol{u}^n|| \leq \bar{u} \quad \forall\, n \tag{15}$$

which are quadratic constraints.

3) *Acceleration constraints:* For the acceleration input, instead of using the nonlinear transform, as shown in (2), we obtain a upper bound for it from the bicycle model

$$a^2(t) = \ddot{x}^2 + \ddot{y}^2 - v^2(t)\dot{\theta}^2 \tag{16a}$$

$$\leq \ddot{x}^2 + \ddot{y}^2. \tag{16b}$$

The intuition is that a is the acceleration in speed instead of velocity. Thus, the change in velocity direction does not contribute to acceleration in our model. The term $v^2(t)\dot{\theta}^2$, which involves heading, is nonnegative. We neglect this term and relax the maximum acceleration constraint as

$$\ddot{x}^2 + \ddot{y}^2 = A_{\cdot,1}^2 + B_{\cdot,1}^2 \leq \bar{a}^2 \tag{17}$$

where $\bar{a}$ represents the maximum acceleration. In practice, if a car is driving in a straight line or at relatively low speed, this relaxation is tight and does not change the original constraint significantly. In our case, since the maximum acceleration $\bar{a}$ is large, the relaxation does not affect the performance of this algorithm.

4) *Curvature constraints:* For the bicycle model, the steering angle is directly related to the geometry of the path and completely determines curvature. Curvature for a plane curve is

$$\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}}.$$

From (2), we have $\phi = \arctan(\kappa L)$. Hence, constraints on steering angle can be transformed to constraints on path

curvature. Simplify the above equation, and the maximum value is

$$\kappa_{\max} = \frac{2(A_{n,2}B_{n,1} - A_{n,1}B_{n,2})}{\min_{t\in[t_n,t_{n+1}]}\left[f(t)\right]^{\frac{3}{2}}} \tag{18}$$

where $f(t)$ is a second-order polynomial in t. These constraints are nonlinear; therefore, we impose the constraint

$$\kappa_{\max} \leq \bar{\kappa}_{\max} \tag{19}$$

using sequential quadratic programming (SQP) [36].

B. Sensitivity-Enhanced Iterated Best Response

The optimization problem that the ego vehicle uses to predict the trajectory of each player is as follows. For player i, the decision variables are $\boldsymbol{A}_i \in \mathbb{R}^{3N}$ and $\boldsymbol{B}_i \in \mathbb{R}^{3N}$, which are the coefficient vectors of N second-order polynomials and $\boldsymbol{\theta}_i = [\mathbf{p}_i^0, \ldots, \mathbf{p}_i^N, \boldsymbol{u}_i^0, \ldots, \boldsymbol{u}_i^N]$, which is consistent with the previous notation. Note that only $\boldsymbol{A}_i$s and $\boldsymbol{B}_i$s are independent and can determine $\boldsymbol{\theta}_i$. We keep $\boldsymbol{\theta}_i$ here for notation clarity. For clarity, rewrite the optimization problem in the following form:

$$\max_{\boldsymbol{\theta}_i, \boldsymbol{A}_i, \boldsymbol{B}_i} \quad s_i(\mathbf{p}_i^N) - s_j(\mathbf{p}_j^N) \tag{20a}$$

$$\text{s.t.} \quad h_i(\boldsymbol{\theta}_i, \boldsymbol{A}_i, \boldsymbol{B}_i) = 0 \tag{20b}$$

$$g_i(\boldsymbol{\theta}_i, \boldsymbol{A}_i, \boldsymbol{B}_i) \leq 0 \tag{20c}$$

$$\gamma_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \leq 0 \tag{20d}$$

where

1) $h_i(\cdot)$ are the equality constraints involving only player i. This includes position and velocity continuity constraints (14).
2) $g_i(\cdot)$ are the inequality constraints involving only player i. This includes track constrains (6), speed constrains (15), acceleration constraints (17), and curvature constrains (19).
3) $\gamma_i(\cdot,\cdot)$ are the inequality constraints involving both players. This include the collision avoidance constraints as in (5).

As in (9), we augment the ego vehicle's objective with the a sensitivity term that reflects the influence on opponents' optimal payoff. The only constraint that involves both players is retained in the sensitivity term (since the ego vehicle cannot influence the other players' strategy directly), and the derivation of the sensitivity analysis to obtain an explicit linear approximation of the objective function is the same as presented in Section III

$$\max_{\boldsymbol{\theta}_i, \boldsymbol{A}_i, \boldsymbol{B}_i} \quad s_i(\mathbf{p}_i^N) + \alpha \boldsymbol{\mu}_j \left.\frac{\partial \gamma_j}{\partial \boldsymbol{\theta}_i}\right|_{(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)} \boldsymbol{\theta}_i. \tag{21}$$

where $\boldsymbol{\theta}_i$, $\boldsymbol{A}_i$, and $\boldsymbol{B}_i$ are subject to constraints (20b)–(20d). Note that α is a tunable parameter in the objective function. The value $\alpha = 0$ means the optimization function does not care about influencing the other players, while larger α leads to more aggressive behavior. We demonstrate the effect of different α parameter choices further in Section V.

The algorithm is summarized in Algorithm 1. The algorithm is solved by the ego vehicle for itself and each other player in sequence until the trajectories converge, always ending with the ego player's optimization problem. We apply the algorithm in receding horizon fashion, and $\mathbf{p}_j^0$ and $\boldsymbol{u}_j^0$ are acquired online. As mentioned in Section III-C, it is possible that multiple Nash equilibria exist in a game. In such cases, different players in a game using the proposed algorithm may converge to different equilibria and result in unsafe trajectories. However, we rarely observe collisions during simulations because of the feedback nature of receding horizon planning approaches. We also note that a low-layer safety controller that operates at higher frequency could be implemented to provide safety guarantee as in [37] and [38].

Algorithm 1: Sensitivity-Enhanced Iterated Best Response.

1: L: maximum number of iterations in IBR
2: decrease schedule of parameter α_{ij}
3: observe opponents' current states $\mathbf{p}_j^0, \boldsymbol{u}_j^0$, $j \neq i$
4: initialize opponents trajectory: $\boldsymbol{\theta}_j^1 = [\mathbf{p}_j^1, \ldots, \mathbf{p}_j^{N+1}]$
5: initialize opponents multiplier: $\boldsymbol{\mu}_{ji}$, $j \neq i$
6: ego: solve (21) using SQP with $\boldsymbol{\theta}_j = \boldsymbol{\theta}_j^1$
7: obtain ego optimal strategy $\boldsymbol{\theta}_i^1$
8: **for** $l = 1, 2, \ldots, L$, or **not converged do**
9: **for** opponent $j \neq i$ **do**
10: opponent: solve (21) using SQP with $\boldsymbol{\theta}_i = \boldsymbol{\theta}_i^l$
11: and $\boldsymbol{\theta}_k = \boldsymbol{\theta}_k^l$, $k \neq i, j$
12: obtain opponent optimal strategy $\boldsymbol{\theta}_j^{l+1}$
13: **end for**
14: ego: solve (21) using SQP with $\boldsymbol{\theta}_j = \boldsymbol{\theta}_j^l$
15: obtain ego optimal strategy $\boldsymbol{\theta}_i^{l+1}$
16: **end for**

Remark 1: One possible extension of this algorithm with lower computation cost is proposed in [4], where lines 9–14 in Algorithm 1 are solved in parallel instead of sequentially. See [4] for analysis on computation speedup and empirical results.

Remark 2: As in other iterative-type nonlinear optimization approaches, the convergence speed and final result of our algorithm are affected by the initialization scheme. In our simulations, we initialize $\boldsymbol{\theta}$ with dynamics propagation from the current state. We found this initialization to be robust and offers good convergence.

V. Simulation Results

The performance of the proposed approach is first validated in simulations. Note that although we use the kinematic bicycle model in the planner, our high-fidelity simulator employs full dynamics of the car with the Fiala tire model [39], [40] and carefully calibrated system parameters to match our experiment vehicle.

The proposed algorithm is implemented using the Gurobi[1] solver with the C++ interface. The nonconvex constraints and objective function are imposed through sequential linearizations. Linearization is applied based on solution from the previous iteration. For the highly nonlinear curvature constraints (19),

[1][Online]. Available: http://www.gurobi.com/

we choose only to enforce the constraints on the first half of the trajectory. We found that this significantly improves the stability of solution. Since the algorithm is implemented in receding horizon fashion, only the first several steps are executed, so the second half of the trajectory will not be executed during the process. Moreover, to achieve online and real-time planning, we solve the optimization problem outlined in Algorithm 1 for two game iterations ($L = 2$) and do not wait until convergence. For both simulation and experiments, we use the robot operating system (ROS) framework, which handles message passing.

We want to highlight three points with simulations in this section. First, we show the advantages of the proposed algorithm competing with a baseline MPC planner in two-car competitions. Second, with two-car competitions, we demonstrate the effect of parameter choice α value in sensitivity-enhanced objective function (9). Finally, we present and discuss the results for three-car competitions.

A. Two-Player Competitions

We first demonstrate the performance of the proposed approach in two-car competing games. The two players use the proposed GTP and a naive MPC planner, respectively.

The naive model-predictive controller serves as a baseline strategy and only optimizes its own objective (8) using an initial guess of the opponent's behavior. In other words, we first initialize the opponent's trajectory $\boldsymbol{\theta}_j$ and then solve (8) with all constraints. The optimal solution is used as the MPC trajectory. Unlike GTP, the baseline MPC planner does not exploit the sensitivity-enhanced term to take advantage of opponent's reaction. And we only solve (8) once instead of iteratively.

We use the following parameters. The racing track is oval-shaped with total length $l_{\boldsymbol{\tau}} = 216$ m and half-width $w_{\boldsymbol{\tau}} = 6.5$ m. The planning horizon is 5 s and the planner updates at 2 Hz. The maximum acceleration for both cars is 5 m/s^2 and the maximum curvature is 0.11 m^{-1}, corresponding to a maximum steering angle of 18°.

We demonstrate the performance of the proposed planner in two settings: overtaking and blocking. In simulations, the two cars start from nominal points 1 and 2, as shown in Fig. 4, with random noise, one always in front of the other. The maximum speed of the leading car is always set to a lower speed limit (5 m/s) than its competitor (6 m/s). The GTP car starts in the leading position in blocking tests and the following position in overtaking tests.

1) In blocking scenarios, the GTP car has the lower maximum speed and thus is in a disadvantageous situation. Naturally, we would expect the slower car to be overtaken. However, in most cases, it is able to block the (higher maximum speed) MPC car and maintain a leading position in the game. Snapshots during a simulation are shown in Fig. 6. Red trajectories are GTP planner trajectories and green trajectories are MPC planner trajectories. Since we adopt a receding horizon planning approach, the illustrated trajectories are planned trajectories instead of true path. We observe that on the first half of the straight segment, the GTP car tends to steer in front of its competitor instead of going straight forward. Even though going straight would achieve more progress along the track for itself, blocking is a better strategy to win the race. When the two cars are entering corners, however, they mostly choose the shortest course as optimal.

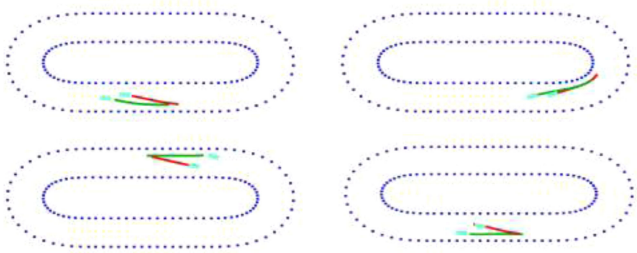


Fig. 6. Snapshots during a simulation in the blocking scenario. The GTP car (red trajectories) starts in the leading position with lower speed limit with respect to the MPC car (green trajectories). The top right figure shows that the two cars entering a corner choose the shortest course. The other three figures show that the leading vehicle is actively blocking the following vehicle by moving intentionally in front of it. This may appear to be suboptimal, but it ensures its leading position in the game.

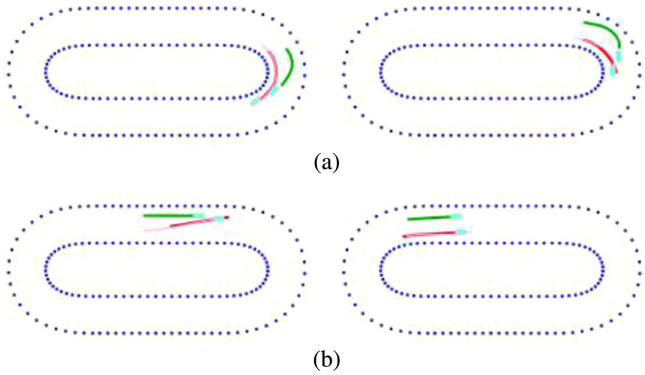


Fig. 7. Snapshots during simulations in the overtaking scenario. The GTP car (red trajectories) starts in the following position with a higher speed limit behind the MPC car (green trajectories). Only key snapshots during overtaking are shown here. (a) Overtaking scenario 1; the MPC car passively avoids collision and steers away from its fastest trajectory. (b) Overtaking scenario 2; the GTP car overtakes the MPC car from the left.

2) In the overtaking scenario, the GTP car uses the higher speed limit and starts slightly behind the MPC car. Note that this is the same configuration as in the blocking scenarios, except that we flip the planning strategies of the two cars. Snapshots during simulations are shown in Fig. 7. Since the MPC car conservatively avoids collision with its opponent, we observe that it may steer out of its fastest path and give way to the other car [see Fig. 7(a)]. In Fig. 7(b), the GTP car overtakes the MPC car on the left. As opposed to the blocking behavior of the GTP car in blocking scenarios, the MPC car continues on its own line and loses the game.

We conducted 100 simulations for each scenario. The starting positions of the two cars are uniformly distributed in two square regions one in front of the other. Each race consists of two laps around the track and ends when either of the two players reaches

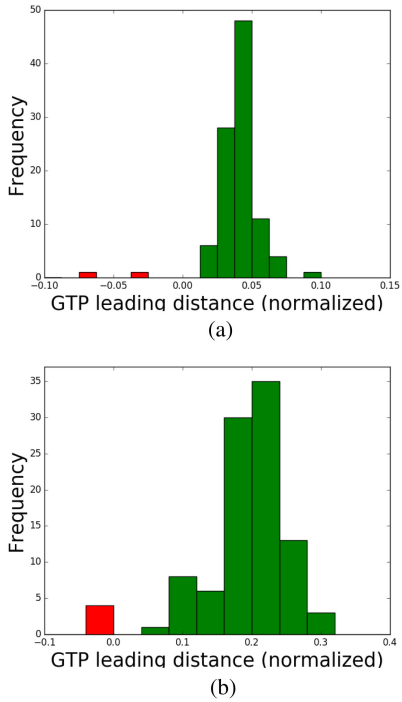


Fig. 8. Histogram plots for blocking and overtaking scenarios. The normalized arc-length distance difference is recorded at the end of each simulation. The bins are colored green if the GTP car finishes first and red otherwise. In both plots, we can tell that the GTP car wins most of the time. (a) Blocking scenario. (b) Overtaking scenario.

the finish line. We recorded the arc-length difference along the track at the end and normalize the arc length with respect to the total length of the track. The histogram results are given in Fig. 8. The bins are colored green if the GTP car finishes the race first and red otherwise. We can see from Fig. 8(a) that when the GTP car has a lower speed limit, it still wins the race 98% of the time since it intentionally plans its trajectory in front of the MPC car and thus blocks its way. The MPC car finishes the race right behind it. In Fig. 8(b), when the GTP car starts behind the MPC car, it is able to overtake and finish the race well before its opponent since it has a higher speed.

B. Effect of Different α Value Choices

In this section, we discuss an important question of how to choose parameter α_i in the sensitivity-enhanced objective function (9). For simplicity, we assume $\alpha_{ij} = \alpha_i,\ \forall j$. The extension to the general case is quite straightforward. With the above assumption, the objective function is reduced to

$$s_i(\boldsymbol{\theta}_i) - \alpha_i \sum_{j \neq i} s_j^*(\boldsymbol{\theta}_i). \tag{22}$$

From this, we can see that α_i value has the intuitive interpretation of balancing player i's own objective and influencing other players' objective. When $\alpha_i = 0$, player i does not leverage its influence on other reactive players, since we are only left with the first term $s_i(\boldsymbol{\theta}_i)$. As α_i increases, player i is rewarded more for penalizing others. Thus, α_i could be interpreted as how *competitive* an agent is. Higher α means more competitive.

TABLE I
PERCENTAGE OF OVERTAKING DURING COMPETITION; FOR DIFFERENT α_i PAIRS, THE PERCENTAGE OF SIMULATIONS WHERE PLAYER 1 (SLOWER) IS OVERTAKEN BY PLAYER 2 (FASTER)

α_1 / α_2	0.5 / 0.1	0.3 / 0.3	0.1 / 0.5
overtaking percentage	0.40	0.86	0.9

To see the effect of α value choices, we conduct simulations with two players, both of them using GTP for trajectory planning but with different α_i. The rest of simulation parameters are the same as in Section V-A. We also use the same random initial positions for the two players. Table I shows the result from simulations. For each α value pair, we run 50 simulations and show the percentage of simulations where player 1 is overtaken by player 2. If α_1 is relatively small, we expect that the "competitiveness" of player 1 is low, and thus, it is more likely to be overtaken. As α_1 increases (relative to player 2), the chance of it being overtaken gets lower. The simulation result matches our expectation and interpretation of α. This again demonstrates that α is a tunable parameter, where a system designer can use to change the behavior of an autonomous competing car.

C. Three-Player Competitions

We then apply the SE-IBR algorithm to three-player competitive games. Among the three players, one player uses the proposed GTP and the two other players use the naive MPC planners. Similar to the two-player games, the MPC planner only avoids collisions passively and does not consider the rich game dynamics among the players.

As for the simulation specifications, the racing track is the same oval-shaped track as in the previous section with total length $l_{\boldsymbol{\tau}} = 216$ m and half-width $w_{\boldsymbol{\tau}} = 10$ m to accommodate more cars. The planning horizon is 5 s and the planner updates at 2 Hz. The maximum acceleration and curvature are the same as in the previous section.

In simulations, the three cars start from positions 1 (Player 1), 2 (Player 2), and 3 (Player 3) as denoted in Fig. 4 with random noise. The maximum speeds for Players 1, 2, and 3 are 5, 5.5, and 6 m/s, respectively. The GTP player starts in position 1 in the blocking scenario and position 3 in the overtaking scenario.

1) In blocking scenarios, the GTP car is incentivized to block the two opponents from moving forward. As can be observed in Fig. 9(a), the GTP could deviate from the optimal path and intent to drive in front of other cars to stay in the first position of the game. Since the GTP car has two opponents, it is also observed that the GTP car may switch between blocking either one of the opponents during the interactions of all three players. This is because of the sensitivity-enhanced term in our optimization problem is only active if the corresponding collision avoidance constraints are active.
2) In overtaking scenarios, the GTP car takes the third position and starts from a disadvantageous position. However, as shown in Fig. 9(b), the slower MPC cars are not able to keep the leading positions. The GTP car, on the other hand,

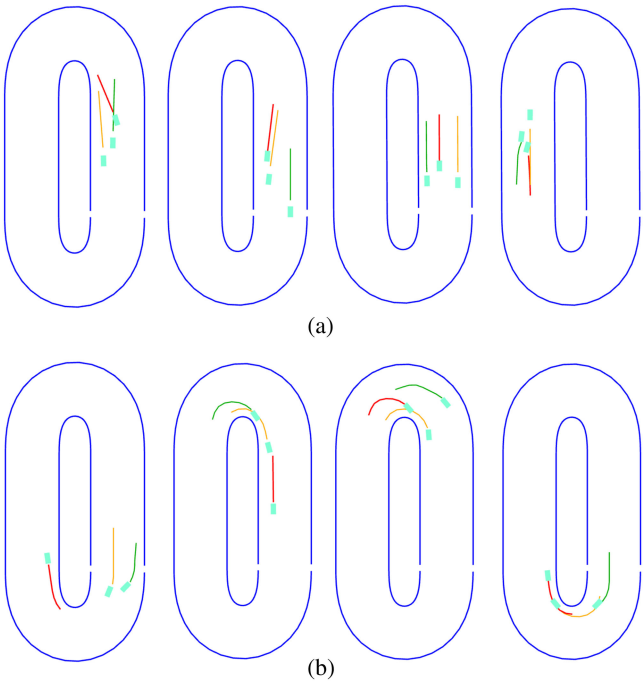


Fig. 9. Snapshots during three-car competitive game simulations. The planned trajectories of the GTP car are colored red in (a) and yellow in (b). The other two cars use naive MPC planners. (a) Blocking scenario. (b) Overtaking scenario.

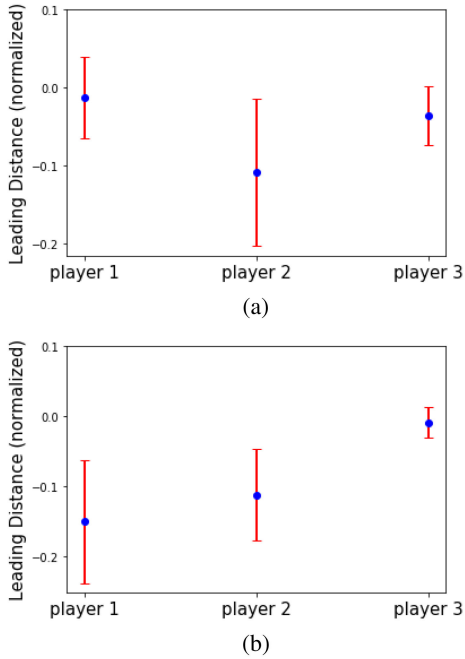


Fig. 10. Statistics results for the two scenarios with three cars competing. Player 1 is the slowest car, while Player 3 is the fastest car. Y-axis of the plots are the leading distance at the end of each game. Higher leading distance is better. The blue dots are the *mean* values of leading distance. The red bars show one *standard deviation* of the data. In (a), player 1 is GTP, while the other two are MPC. In (b), player 3 is GTP, and the other two are MPC. (a) Blocking scenario. (b) Overtaking scenario.

exploits the dynamic game interactions and overtakes the opponents.

We run 50 trials for each scenario. In each trial, the three cars start from their starting positions with maximum speed. A trial ends when any of them finishes two laps at the end line. We recorded the arc-length difference along the track at the end of each trial and normalize it by total track length. The first car to finish has leading distance 0. The other cars have negative leading distance. A statistical result is shown in Fig. 10.

For the blocking case, we can see that the GTP car (Player 1) has the highest (mean) leading distance, thus showing that the GTP can successfully win the game despite having a slower maximum speed. The mean value of Player 1 is negative since it might be overtaken by faster cars. From the statistics of Players 2 and 3, Player 3 has a higher mean leading distance since it has higher speed. The performance of Player 2 has more stochasticity. It could either overtake Player 1 in rare cases or overtaken by Player 3 in other cases. We also notice that the results of two MPC players have more randomness, and the final outcome depends on the complex interactions between all three agents. For the overtaking case, our proposed planner also significantly outperforms the other two cars with the highest leading distance.

VI. Experimental Results

A. Small-Scale Car Experiments

We first conducted experiments using a small-scale autonomous car platform based on a hobby RC car kit, with the Optitrack motion capture system.

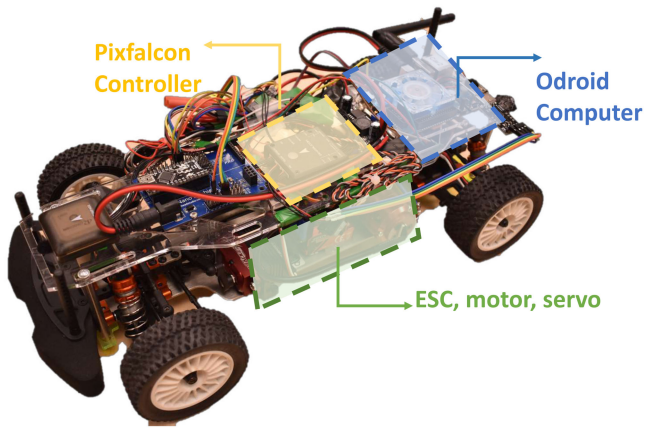


Fig. 11. 1/10 scale RC car experimental platform.

For indoor experiments, we use two 1/10 scale RC cars equipped with Pixfalcon flight controllers for low-level control, such as steering and motor control. The RC cars also have Odroid-XU4 computers and Electronic Speed Controllers (ESC) onboard. The system structure of an RC car is shown in Fig. 11. The physical dimension of the cars are 20 cm wide and 40 cm long. An Optitrack motion capture system broadcasts pose information of both cars at 100 Hz, which is further used by a low-pass filter to estimate velocity. State information of both cars is passed to a DELL XPS laptop with dual-core 2.7-GHz Intel i7-7500 U

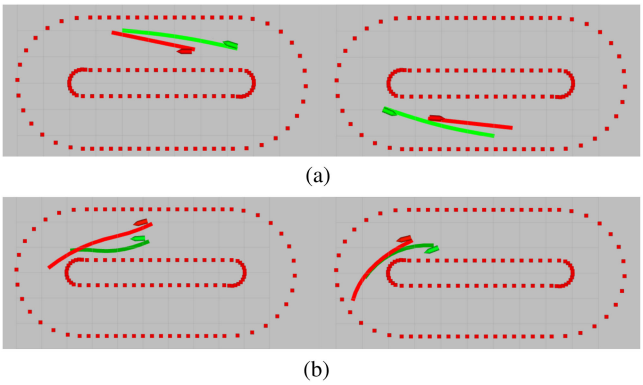


Fig. 12. Snapshots during RC car experiments. The red trajectories are for the GTP car and the green trajectories are for the MPC car. In (a), the blocking scenario is shown where the GTP car adopts a trajectory in front of its opponent instead of going straight. In (b), the GTP car aggressively drives pass the MPC car to overtake, knowing that the MPC car would slow down to avoid a collision. (a) RC car blocking scenario visualization. (b) RC car overtaking scenario visualization.

processor. The laptop runs two planners for the two cars separately and publishes desired trajectory information to two cars. Pure-pursuit controllers are used to generate control commands for trajectory tracking. The racing track is oval-shaped and has two straight sections at 6 m and curved sections with radius 1.5 m. Track half-width is 0.5 m. The high and low speed limits are 2.5 and 1.8 m/s, respectively. The planning horizon is 2 s and the planners update at 2 Hz. The relative position at the start of the race is set to be the same as in simulations. We did ten experiments of blocking and the GTP car won the races in all tests. We also conducted ten experiments of overtaking, out of which nine were successful. Each experiment consists of two laps in the blocking scenario or one successful overtaking. Snapshots of blocking and overtaking experiments are given in Fig. 12.

B. Full-Sized Car Experiments

We also conducted experiments on X1, a student-built research vehicle. X1 is a student-built research vehicle, as shown in Fig. 2. X1 is a four-wheel steer-by-wire, brake-by-wire, electrically driven vehicle. It is equipped with differential global positioning system (GPS) and inertial navigation system for precise position measurements. A low-level real-time processor takes steering and acceleration commands at 100 Hz and sends low-level commands to the steering actuators and electric motor. X1 makes use of the MPC algorithm developed in [41] to ensure safe handling while tracking a reference path. Using the ROS framework, the game-theoretic controller sends a reference path to the low-level path tracking controller and receives vehicle state information in return. Due to safety considerations, we opt for running X1 along with a simulated car, rather than with real cars. To simulate the sensor measurements of the position of the other car, we send the GPS or simulated GPS measurement to the opponent through ROS. Other than this, the GTP of the test vehicle and the simulated car run independently without any communication. The test cases and parameters are the same, as depicted in Section V. The test vehicle is always using the GTP and the simulated car is using an MPC planner.

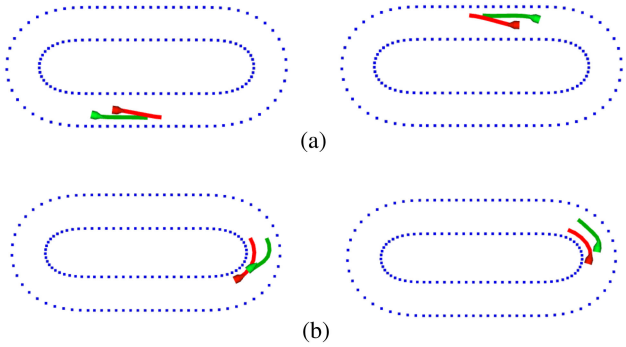


Fig. 13. Snapshots of X1 versus a simulated car experiments. As before, the red trajectories are for the GTP car and the green trajectories are for the MPC car. (a) shows the blocking behavior by the proposed planner and (b) shows overtaking while the MPC car gives way. (a) X1 blocking scenario visualization. (b) X1 overtaking scenario visualization.

Snapshots plotted using recorded GPS data during the experiments are shown in Fig. 13. In the blocking scenario [see Fig. 13(a)], we did ten experiments (20 laps) with GTP successfully blocking the other car. During the corners, both cars will go close to the inner boundary since it is more advantageous to take the shorter course. In the overtaking scenarios [see Fig. 13(b)], we did seven experiments of successful overtaking. We can see that the MPC car does not actively block its opponent as compared to the behavior of the GTP car in the previous scenario. In fact, the MPC car give way to the GTP car in some experiments. The reason is that when the opponent's predicted future trajectory interferes with its own trajectory, it passively avoids the other car and get out of their way.

VII. Conclusion

In this article, we presented an online game-theoretic trajectory planner based on the concept of Nash equilibria. A modified SE-IBR algorithm was used to solve for the approximate Nash equilibrium in the space of feasible trajectories for car-like robot kinematics. We leveraged the differential flatness property of the bicycle model in order to satisfy the car's acceleration and curvature constraints when solving for the competitive trajectories. The planner was shown to be suitable for online planning and exhibits complex competitive behaviors in racing scenarios with multiple cars. The performance of the proposed planner was demonstrated in both simulation and experiments with different vehicle hardware platforms with different scales.

Limitations and future work: While our simulations and experiments show that the GTP framework outperforms baseline MPC planners in competitive racing scenarios, a key future direction is its application in more realistic traffic scenarios. In traffic scenarios, such as highway merging, lane changing, or navigating an intersection, the players does not necessarily have conflicting objectives and competitive–cooperative games are more suitable for such cases. A few recent works address this problem by modeling a general game framework either

with explicit constraints [42] and use augmented Lagrangian-type solutions or penalize constraints in objective functions and use an iterative scheme as in [23]. Furthermore, while we constraint the problem to have full deterministic information on models and objective functions of all game players, future work should consider removing these assumptions and to learn a reward function or infer the vehicles intention online. These extensions are by itself active research areas. Another direction that should be explored is the uncertainty during interactions. For example, the authors of [43] and [44] address the problem of perception system during interactions. We recently proposed a risk-sensitive game-theoretic planning approach to model the risk-sensitive behavior of traffic participants while considering game interactions [45]. Risk-sensitive planning is also important for application in real traffic beyond considering the feedback interactions.

REFERENCES

[1] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game theoretic planning for self-driving cars in competitive scenarios," in *Proc. Robot.: Sci. Syst. Conf.*, Jun. 2019.

[2] 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. 36, no. 5, pp. 1389–1403, 2020.

[3] Z. Wang, R. Spica, and M. Schwager, "Game theoretic motion planning for multi-robot racing," in *Distributed Autonomous Robotic Systems*. Berlin, Germany: Springer, 2019, pp. 225–238.

[4] 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," *Robot. Auton. Syst.*, vol. 125, Mar. 2020, Art. no. 103410.

[5] 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 *Proc. IEEE Int. Conf. Robot. Autom.*, May 2018, pp. 2403–2410.

[6] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Proc. Robot.: Sci. Syst. Conf.*, Jun. 2016.

[7] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 9590–9596.

[8] S. M. Thornton, F. E. L. V. Zhang, M. J. Kochenderfer, and J. C. Gerdes, "Value sensitive design for autonomous vehicle motion planning," in *Proc. IEEE Intell. Vehi. Symp.*, Jun. 2018, pp. 26–30.

[9] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[10] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transp. Res. C, Emerg. Technol.*, vol. 60, pp. 416–442, 2015.

[11] G. Ding, S. Aghli, C. Heckman, and L. Chen, "Game-theoretic cooperative lane changing using data-driven models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Oct. 2018, pp. 3640–3647.

[12] J. H. Yoo and R. Langari, "A Stackelberg game theoretic driver model for merging," in *Proc. ASME Dyn. Syst. Control Conf.*, Oct. 2013, vol. 56130, Paper V002T30A003.

[13] A. Dreves and M. Gerdts, "A generalized Nash equilibrium approach for optimal control problems of autonomous cars," *Optimal Control Appl. Methods*, vol. 39, no. 1, pp. 326–342, 2018.

[14] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 5, pp. 1782–1797, Sep. 2018.

[15] M. Bahram, A. Lawitzky, J. Friedrichs, M. Aeberhard, and D. Wollherr, "A game-theoretic approach to replanning-aware interactive scene prediction and planning," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3981–3992, Jun. 2016.

[16] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[17] A. Liniger and J. Lygeros, "A noncooperative game approach to autonomous racing," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 3, pp. 884–897, May. 2020.

[18] A. Turnwald, D. Althoff, D. Wollherr, and M. Buss, "Understanding human avoidance behavior: Interaction-aware decision making based on game theory," *Int. J. Social Robot.*, vol. 8, no. 2, pp. 331–351, 2016.

[19] Z. Zhou, R. Takei, H. Huang, and C. J. Tomlin, "A general, open-loop formulation for reach-avoid games," in *Proc. 51st IEEE Conf. Decis. Control*, 2012, pp. 6501–6506.

[20] J. S. Jang and C. Tomlin, "Control strategies in multi-player pursuit and evasion game," in *Proc. AIAA Guid., Navigat., Control Conf. Exhib.*, Aug. 2005, Art. no. AIAA 2005-6239.

[21] I. Hwang, D. M. Stipanovic, and C. J. Tomlin, "Applications of polytopic approximations of reachable sets to linear dynamic games and a class of nonlinear systems," in *Proc. Amer. Control Conf.*, Jun. 2003, vol. 6, pp. 4613–4619.

[22] S.-Y. Liu, Z. Zhou, C. Tomlin, and J. K. Hedrick, "Evasion of a team of dubins vehicles from a hidden pursuer," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2014, pp. 6771–6776.

[23] 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 *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 1475–1481.

[24] A. Dreves, "A best-response approach for equilibrium selection in two-player generalized Nash equilibrium problems," *Optimization*, vol. 68, no. 12, pp. 2269–2295, 2019.

[25] A. Dreves, "How to select a solution in generalized Nash equilibrium problems," *J. Optim. Theory Appl.*, vol. 178, no. 3, pp. 973–997, 2018.

[26] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered Voronoi cells," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 1047–1054, Apr. 2017.

[27] M. Wang, Z. Wang, S. Paudel, and M. Schwager, "Safe distributed lane change maneuvers for multiple autonomous vehicles using buffered input cells," in *Proc. Int. Conf. Robot. Autom.*, 2018, pp. 4678–4684.

[28] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. 53rd IEEE Conf. Decis. Control*, 2014, pp. 6271–6278.

[29] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes, "Collision avoidance and stabilization for autonomous vehicles in emergency scenarios," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 4, pp. 1204–1216, Jul. 2017.

[30] K. Leung, E. Schmerling, M. Chen, J. Talbot, J. C. Gerdes, and M. Pavone, "On infusing reachability-based safety assurance within probabilistic planning frameworks for human-robot vehicle interactions," in *Proc. Int. Symp. Exp. Robot.*, 2018, pp. 561–574.

[31] M. Van Nieuwstadt, M. Rathinam, and R. M. Murray, "Differential flatness and absolute equivalence of nonlinear control systems," *SIAM J. Control Optim.*, vol. 36, no. 4, pp. 1225–1239, 1998.

[32] R. M. Murray, M. Rathinam, and W. Sluis, "Differential flatness of mechanical control systems: A catalog of prototype systems," in *Proc. ASME Int. Mech. Eng. Congr. Expo.*, 1995.

[33] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, vol. 23. Philadelphia, PA, USA: SIAM, 1999.

[34] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, "The complexity of computing a Nash equilibrium," *SIAM J. Comput.*, vol. 39, no. 1, pp. 195–259, 2009.

[35] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[36] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numer.*, vol. 4, pp. 1–51, 1995.

[37] G. Notomista, M. Wang, M. Schwager, and M. Egerstedt, "Enhancing game-theoretic autonomous car racing using control barrier functions," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 5393–5399.

[38] K. Leung *et al.*, "On infusing reachability-based safety assurance within planning frameworks for human-robot vehicle interactions," *Int. J. Robot. Res.*, vol. 39, pp. 1326–1345, 2020.

[39] H. Pacejka, *Tire and Vehicle Dynamics*. Amsterdam, The Netherlands: Elsevier, 2005.

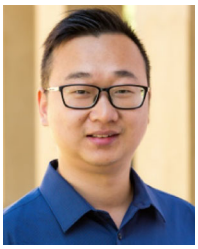
[40] H. B. Pacejka, *Tire Characteristics and Vehicle Handling and Stability*. Oxford, U.K.: Butterworth-Heinemann, Dec. 2012, pp. 1–58.

[41] M. Brown, J. Funke, S. Erlien, and J. C. Gerdes, "Safe driving envelopes for path tracking in autonomous vehicles," *Control Eng. Pract.*, vol. 61, pp. 307–316, 2017.

[42] S. L. Cleach´, M. Schwager, and Z. Manchester, "ALGAMES: A fast solver for constrained dynamic games," in *Proc. Robot.: Sci. Syst. Conf.*, Jun. 2020.
[43] J. Yoo and R. Langari, "A predictive perception model and control strategy for collision-free autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 11, pp. 4078–4091, Nov. 2019.
[44] Q. Zhang, R. Langari, H. E. Tseng, D. Filev, S. Szwabowski, and S. Coskun, "A game theoretic model predictive controller with aggressiveness estimation for mandatory lane change," *IEEE Trans. Intell. Veh.*, vol. 5, no. 1, pp. 75–89, Mar. 2020.
[45] M. Wang, N. Mehr, A. Gaidon, and M. Schwager, "Game-theoretic planning for risk-aware interactive agents," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020.

Mingyu Wang received the B.S. degree in control science and engineering from the Harbin Institute of Technology, Harbin, China, in 2015, and the M.S. degree in mechanical engineering in 2020 from Stanford University, Stanford, CA, USA, where she is currently working toward the Ph.D. degree in mechanical engineering.

Her research interests include game-theoretic planning, optimization, and control, especially for autonomous cars interacting with other human-driven or autonomous cars.



Zijian Wang received the B.S. degree in automation and mechatronic engineering from Beihang University, Beijing, China, in 2013, the M.S. degree in mechanical engineering from Boston University, Boston, MA, USA, in 2016, and the Ph.D. degree in aeronautics and astronautics from Stanford University, Stanford, CA, USA, in 2019.

His current research interests include multirobot systems, particularly on designing distributed planning and control algorithms that enable a group of intelligent robots to either collaborate on a common task, or compete among adversarial agents.

John Talbot received the B.M.E. degree in mechanical engineering from Auburn University, Auburn, AL, USA, in 2016, and the M.S. degree in mechanical engineering in 2019 from Stanford University, Stanford, CA, USA, where he is currently working toward the Ph.D. degree in mechanical engineering.

His current research interests include aiding human drivers in difficult driving scenarios at the limits of vehicle handling using nonlinear model-predictive control.

J. Christian Gerdes (Member, IEEE) received the Ph.D. degree in mechanical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 1996.

He is currently a Professor of Mechanical Engineering with Stanford University, Stanford, CA, USA, and the Director of the Center for Automotive Research at Stanford, Stanford University. He is a Co-Founder of Peloton Technology, Mountain View, CA. His laboratory studies how cars move, how humans drive cars, and how to design future cars that work cooperatively with the driver or drive themselves. When not teaching on campus, he can often be found at the racetrack with students, instrumenting historic race cars, or trying out their latest prototypes for the future.

Prof. Gerdes and his team have been recognized with several awards, including the Presidential Early Career Award for Scientists and Engineers, the Ralph Teetor Award from the SAE International, and the Rudolf Kalman Award from the American Society of Mechanical Engineers.

Mac Schwager (Member, IEEE) received the B.S. degree from Stanford University, Stanford, CA, USA, in 2000, and the M.S. and Ph.D. degrees from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2005 and 2009, respectively, all in mechanical engineering.

He is currently an Assistant Professor with the Department of Aeronautics and Astronautics, Stanford University. He was a Postdoctoral Researcher working jointly with the GRASP Lab, University of Pennsylvania, Philadelphia, PA, USA, and Computer Science and Artificial Intelligence Laboratory, MIT, from 2010 to 2012. He was an Assistant Professor with Boston University, Boston, MA, USA, from 2012 to 2015. His research interests include distributed algorithms for control, perception, and learning in groups of robots, and models of cooperation and competition in groups of engineered and natural agents.

Dr. Schwager was the recipient of the NSF CAREER Award in 2014, the DARPA Young Faculty Award in 2018, the Google Faculty Research Award in 2018, and the IROS Toshio Fukuda Young Professional Award in 2019.