Robotics: Science and Systems 2019
Freiburg im Breisgau, June 22-26, 2019

Game Theoretic Planning for Self-Driving Cars in Competitive Scenarios

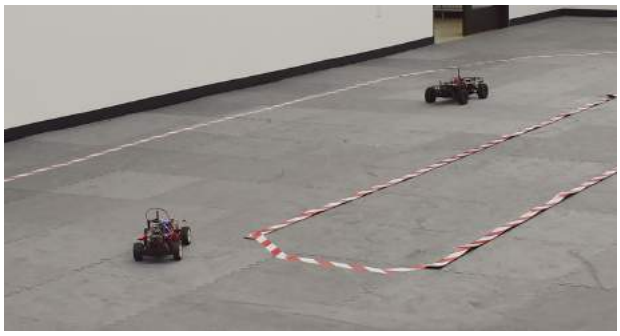
Mingyu Wang[1], Zijian Wang[2], John Talbot[1], J. Christian Gerdes[1] and Mac Schwager[2]

***Abstract*—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 two car autonomous racing game in which each car tries to advance along a given track as far as possible with respect to the other car. The algorithm extends previous work on game-theoretic planning for single integrator agents to be suitable for autonomous cars in the following ways: (i) by representing the trajectory as a piecewise-polynomial, (ii) incorporating bicycle kinematics into the trajectory, (iii) enforcing constraints on path curvature and acceleration. The game theoretic planner iteratively plans a trajectory for the ego vehicle, then the other vehicle until convergence. Crucially, the trajectory optimization includes a sensitivity term that allows the ego vehicle to reason about how much the other vehicle 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 out-perform a baseline planner using Model Predictive Control which does not take interaction into account. The performance is validated in high-fidelity numerical simulations, in experiments with two scale autonomous cars, and in experiments with a full-scale autonomous car racing against a simulated vehicle.**

I. Introduction

In this work we seek to enable an autonomous car to interact with other vehicles, both human-driven and autonomous, while reasoning about the other vehicle's 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 found in numerous 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 two car racing game in which the objective of each car is to advance along the track as far beyond its competitor as possible. The vehicles also share a common collision avoidance constraint to formalize the intention that both cars want to avoid collisions. Nevertheless, there is no means of explicit communication between the vehicles. It is through this collision avoidance constraint that one agent can impose influence on the other, 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 the two cars. The ego vehicle iteratively



(a) Indoor experiments using RC car experimental platform. Motion capture system is used for odometry information of both cars.

(b) Outdoor experiments using X1 — a drive-by-wire research vehicle at Thunderhill Raceway Park, CA

Fig. 1: We demonstrate the performance of our algorithm in experiments using both RC cars indoors and a full-size research vehicle, X1, outdoors.

[1]Mingyu Wang, John Talbot, and J. Christian Gerdes are with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA, {mingyuw, john.talbot, gerdes}@stanford.edu.

[2]Zijian Wang and Mac Schwager are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA, {zjwang, schwager}@stanford.edu.

Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

solves several rounds of trajectory optimization, first for itself, then for the opponent, then again for itself, etc. The fixed point of this iteration represents a Nash equilibrium in the trajectories of the two vehicles, hence the trajectory for the ego vehicle accounts for the predicted response of the other vehicle. The planned trajectories are specifically suited to cars as we parameterize them by piecewise time polynomials which take into account the vehicle kinematics and also can enforce the bounded steering and acceleration constraints.

We verify our algorithm in both simulations and experiments. The simulations use a high-fidelity model of vehicle dynamics and show that our game theoretic planner significantly outperforms an opponent using a conservative MPC-based trajectory planner which only avoids collisions passively. We also perform experiments with scale model cars, and a full sized autonomous car. For the model cars, we show again that the car using our game theoretic planner significantly out-performs its competitor with the aforementioned MPC based planner. A snapshot from an experimental run is shown in Fig. 1a. For the full-scale car (shown in Fig. 1b), we show that the game theoretic planner outperforms a simulated competing vehicle (simulated for safety reasons) using the aforementioned MPC-based planner.

II. Related Work

Our work builds upon the results in [1], in which the Sensitivity Enhanced Iterated Best Response (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 [2], [3], Wang et al. extended this algorithm to multiple drones (more than two) in 3d environments, again with single integrator dynamics. The main innovation in our work beyond these is (i) to represent the trajectory as a continuous-time piecewise polynomial, (ii) to incorporate bicycle kinematic constraints, (iii) and 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 lab, and experimental results with a full-scale autonomous car on a test track.

Some other recent works have proposed Iterative Best Response (IBR) algorithms to reach Nash equilibria in multi-vehicle interaction scenarios. In [4], Williams et al. propose an IBR algorithm together with an information theoretic planner for controlling two ground vehicles in close proximity. That work considers problems in which the objective function of each vehicle depends explicitly on the states of both vehicles. In contrast, in our problem, 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. Secondly, the scenario in [4] is collaborative: the vehicles try to maintain a prescribed distance from one another. Our scenario is competitive: the vehicles are competing to win a race. Finally, the algorithm in [4] 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 [5] proposes 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.

Aside from the above examples, the existing literature in game-theoretic motion planning and modeling for multiple competing agents typically are either for discrete state or action spaces [6]–[8], or are specifically for pursuit-evasion style games. In [9], a hierarchical reasoning game theory approach is used for interactive driver modeling; [10] predicts the motion of vehicles in a model-based, intention-aware framework using an extensive-form game formulation; [11], [12] considers Nash and Stackelberg equilibria in a two-car racing game where the action space of both players are 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 [13]. 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 [14]–[17] in the context of a pursuit-evasion game where pursuers and evaders are antagonistic towards each other. This antagonistic assumption is unrealistic in real-world driving and leads to over conservative behaviors for the ego vehicle.

There is also a vast body of work on collision avoidance in a non-game-theoretic framework. To name just a few, [18], [19] present a provable distributed collision avoidance framework for multiple robots using Voronoi diagrams, under the assumption that agents are reciprocal. Control barrier functions are used in [20] for adaptive cruise control. [21] presents a control structure for cars in emergency scenarios where cars have to maneuver at their limits to achieve stability and avoid collisions with static obstacles. [22] introduces a minimally-interventional controller for closed-loop collision avoidance using backward reachability analysis. By contrast, we focus on motion planning in a competitive two-car racing game, where agents are adversarial and have influence on each other through shared collision avoidance constraints.

III. Preliminaries

In this section, the game theoretic planner from [1] is concisely summarized for two simplified discrete-time single-integrator robots in a racing scenario. Our work in this paper builds upon this previously proposed framework.

A. Discrete Time Single Integrator Racing Game

Consider two single integrator agents competing in a racing scenario. To simplify the control strategy, we assume they move in $\mathbb{R}^2$ with the following discrete time dynamics:

$$\mathbf{p}_i^n = \mathbf{p}_i^{n-1} + \mathbf{u}_i^n,$$

where $\mathbf{p}_i^n \in \mathbb{R}^2$ and $\mathbf{u}_i^n \in \mathbb{R}^2$ are robot i's position and velocity at time step n, respectively. We assume we have direct control over velocity so $\mathbf{u}_i^n$ is also our control input. The racing track is characterized by its center line:

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

where $l_{\boldsymbol{\tau}}$ is total track length. Track tangent and normal vectors are denoted as: $\boldsymbol{t} = \boldsymbol{\tau}'$ and $\boldsymbol{n} = \boldsymbol{\tau}''$. Track width is $w_{\boldsymbol{\tau}}$. These track parameters are known by all racing agents beforehand. The progress of one player 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:

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

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

$$\max_{\boldsymbol{\theta}_i} \; s_i(\mathbf{p}_i^N) - s_j(\mathbf{p}_j^N) \tag{2a}$$

$$\text{s.t.} \quad \mathbf{p}_i^n = \mathbf{p}_i^{n-1} + \mathbf{u}_i^n \tag{2b}$$

$$\left\|\mathbf{p}_j^n - \mathbf{p}_i^n\right\| \geq \overline{d}_i \tag{2c}$$

$$\left|\boldsymbol{n}(\mathbf{p}_i^n)^T[\mathbf{p}_i^n - \boldsymbol{\tau}(\mathbf{p}_i^n)]\right| \leq w_{\boldsymbol{\tau}} \tag{2d}$$

$$\|\boldsymbol{u}_i^n\| \leq \overline{u}_i, \tag{2e}$$

where the decision variable is $\boldsymbol{\theta}_i = [\mathbf{p}_i^1, \ldots, \mathbf{p}_i^N, \boldsymbol{u}_i^1, \ldots, \boldsymbol{u}_i^N]$. The optimization will be solved and used in a receding horizon planning fashion with N steps. (2a) represents the objective of one player to travel further than its opponent at the end of planning horizon. Constraint (2b) enforces the integrator dynamics, (2c) is the collision avoidance constraint for the two players which we denote by i and j, $\overline{d}_i$ is the minimum clearance distance for agent i to avoid collision (may be different or the same for different agents, and can be used to represent risk tolerance for collisions). The constraint (2d) keep the player inside the track boundaries, and (2e) is the actuation limit constraint with maximum velocity $\overline{u}_i$, which may be different for different agents.

In this optimization problem, note that player i only has access to its own planned trajectory $\boldsymbol{\theta}_i$. The opponent's planned trajectory $\boldsymbol{\theta}_j$ is not available. However, because of the collision avoidance constraints (2c), the ego player needs $\mathbf{p}_j$s to calculate its optimal solution. In other words, the feasible trajectory set of player i depends on the unknown trajectory of player j.

To deal with the competitive nature of this problem, we model it as a zero-sum game, which could be seen explicitly by writing the objective of player j as $s_j(\mathbf{p}_j^N) - s_i(\mathbf{p}_i^N)$. We use the concept of Nash equilibria [23]. Unlike in [5], where the robot car is assumed to take action first and the human driven car responds, in our work, no player has information advantage over its opponent. Thus, Nash equilibria better suit the problem. A Nash equilibrium is a strategy profile (a trajectory pair, in our case) where no player can do better by unilaterally changing its strategy. Formally, $\Theta = \Theta_i \times \Theta_j$ is the set of feasible strategy pairs for both players. $\Theta_i(\boldsymbol{\theta}_j)$ is the set of feasible strategies of player i given the strategy of player j. $f_i(\boldsymbol{\theta})$ and $f_j(\boldsymbol{\theta})$ are the payoff functions for player i and j evaluated at $\boldsymbol{\theta} \in \Theta$. A strategy profile $\boldsymbol{\theta}^* = \boldsymbol{\theta}_i^* \times \boldsymbol{\theta}_j^*$ is a Nash equilibrium if

$$\boldsymbol{\theta}_i^* = \arg\max_{\boldsymbol{\theta}_i \in \Theta_i(\boldsymbol{\theta}_j^*)} f_i(\boldsymbol{\theta}_i), \tag{3a}$$

$$\boldsymbol{\theta}_j^* = \arg\max_{\boldsymbol{\theta}_j \in \Theta_j(\boldsymbol{\theta}_i^*)} f_j(\boldsymbol{\theta}_j). \tag{3b}$$

Computing Nash equilibria in general can be computationally intractable and there can be multiple Nash equilibria in a game. Instead, an iterative algorithm can be used to approach a Nash equilibrium online at each time step, as explained below.

B. Sensitivity Enhanced Objective Function

To simplify the notation, (2) can be rewritten as:

$$\max_{\boldsymbol{\theta}_i} \; s_i(\boldsymbol{\theta}_i) - s_j(\boldsymbol{\theta}_j), \tag{4a}$$

$$\text{s.t.} \quad h_i(\boldsymbol{\theta}_i) = 0, \tag{4b}$$

$$g_i(\boldsymbol{\theta}_i) \leq 0, \tag{4c}$$

$$\gamma_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \leq 0, \tag{4d}$$

where $h_i(\cdot)$ represents equality constraint (2b); $g_i(\cdot)$ represents inequality constraints (2e), (2d) and $\gamma_i(\cdot, \cdot)$ is the inequality constraint (2c) involving both players which provides the only coupling between the ego player and its opponent. As we pointed out, although player i does not have control authority of the strategy of player j, it could still pose influence on the calculation of the opponent's optimal strategy through the coupled collision avoidance constraint (2c). In other words, player i's strategy does affect the payoff of player j at Nash equilibria. Thus, the optimal payoff of player j can be represented as a function of $\boldsymbol{\theta}_i^*$, i.e., $s_j^*(\boldsymbol{\theta}_i^*)$. To incorporate the influence of one player on its opponent's optimal payoff, we propose to substitute the objective in problem (4) by:

$$s_i(\boldsymbol{\theta}_i) - \alpha s_j^*(\boldsymbol{\theta}_i). \tag{5}$$

A closed-form solution of $s_j^*(\boldsymbol{\theta}_i)$ is not easily available. However, we could apply sensitivity analysis [24] and get a linear approximation of $s_j^*(\boldsymbol{\theta}_i)$ around an available solution of $\boldsymbol{\theta}_i$. A sensitivity enhanced Iterated Best Response (IBR) algorithm is used in our algorithm. The algorithm starts from an initial guess of opponent's trajectory; solves the optimization problem (4) with sensitivity-enhanced objective (5) for both the ego and opponent player iteratively. When solving the problem for one player, we freeze and use the solution of the other player's strategy from the previous iteration.

Specifically, assume that in the lth iteration, the solution of ego player's optimization problem from last iteration is $\boldsymbol{\theta}_i^{l-1}$. Then, $\boldsymbol{\theta}_i^{l-1}$ is treated as fixed when the ego robot solves the optimization for the opponent to obtain the opponent's optimal value $s_j^*(\boldsymbol{\theta}_i^{l-1})$. If we linearize $s_j^*(\boldsymbol{\theta}_i)$ around $\boldsymbol{\theta}_i^{l-1}$:

$$s_j^*(\boldsymbol{\theta}_i) = s_j^*(\boldsymbol{\theta}_i^{l-1}) + \left.\frac{\mathrm{d}s_j^*}{\mathrm{d}\boldsymbol{\theta}_i}\right|_{\boldsymbol{\theta}_i = \boldsymbol{\theta}_i^{l-1}} (\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{l-1}). \tag{6}$$

It is derived in [1] that:

$$\left.\frac{\mathrm{d}s_j^*}{\mathrm{d}\boldsymbol{\theta}_i}\right|_{\boldsymbol{\theta}_i^{l-1}} = -\boldsymbol{\mu}_j^l \left.\frac{\partial \gamma_j}{\partial \boldsymbol{\theta}_i}\right|_{(\boldsymbol{\theta}_i^{l-1}, \boldsymbol{\theta}_j^l)}, \tag{7}$$

where $\boldsymbol{\mu}_j^l$ is a row vector representing the Lagrange multiplier associated with the inequality constraints. Combine (5), (6), (7) and neglect the constant terms, the optimization problem that each player should solve is:

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

Note that the computation of $s_i(\boldsymbol{\theta}_i)$ also requires an optimization problem as given in (1). Again, a linear approximation of this term by sensitivity analysis is obtained. The final explicit form of the optimization problem is as follows:

$$\max_{\boldsymbol{\theta}_i \in \Theta_i^l} \boldsymbol{t}^T \mathbf{p}_i^N + \alpha \boldsymbol{\mu}_j^l \left.\frac{\partial \gamma_j}{\partial \boldsymbol{\theta}_i}\right|_{(\boldsymbol{\theta}_i^{l-1}, \boldsymbol{\theta}_j^l)} \boldsymbol{\theta}_i. \tag{8}$$

Using this sensitivity-enhanced objective function, it is proved in [1] that if the Sensitivity Enhanced Iterated Best Response iteration converges, then the resulting strategy pair satisfies the necessary conditions for a Nash equilibrium. For a detailed derivation of the results, the readers are referred to [1].

IV. Car Game Theoretic Planner

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

A. Bicycle model of cars

To plan a feasible trajectory for cars, our planner requires a more realistic kinematic motion model. For this purpose, we use the following kinematic bicycle model,

$$\begin{aligned} \dot{x}(t) &= v(t)\cos\theta(t), \quad \dot{y}(t) = v(t)\sin\theta(t), \\ \dot{\theta}(t) &= \frac{v(t)}{L}\tan\phi(t), \quad \dot{v}(t) = a(t), \end{aligned} \tag{9}$$

where x, y, v, θ are 2D position, speed and heading of the car; a and ϕ are acceleration and steering angle commands. L is the distance between front and rear axles. The bicycle model above can be shown to be differentially flat [25], [26] with the flat output $\sigma = [\sigma_1,\ \sigma_2]^T = [x,\ y]^T$. Hence we can analytically compute the control inputs and all state variables given trajectories of $\sigma_1(t)$ and $\sigma_2(t)$ and their derivatives up to second order. Such relationship is called endogenous transformation. For bicycle model, this transformation is given as:

$$\begin{aligned} x &= \sigma_1, & y &= \sigma_2, \\ \theta &= \arctan 2(\dot{\sigma_2}, \dot{\sigma_1}), & v &= \sqrt{\dot{\sigma_1}^2 + \dot{\sigma_2}^2}, \\ a &= \frac{\dot{\sigma_1}\ddot{\sigma_1} + \dot{\sigma_2}\ddot{\sigma_2}}{\sqrt{\dot{\sigma_1}^2 + \dot{\sigma_2}^2}}, & \phi &= \arctan\left(\frac{\dot{\sigma_1}\ddot{\sigma_2} - \dot{\sigma_2}\ddot{\sigma_1}}{(\dot{\sigma_1}^2 + \dot{\sigma_2}^2)^{\frac{3}{2}}} L\right). \end{aligned} \tag{10}$$

Using this endogenous transformation, a trajectory could be planned in the flat output space instead of directly handling the nonlinear dynamic constraints. Control inputs could be derived from flat outputs afterwards using (10).

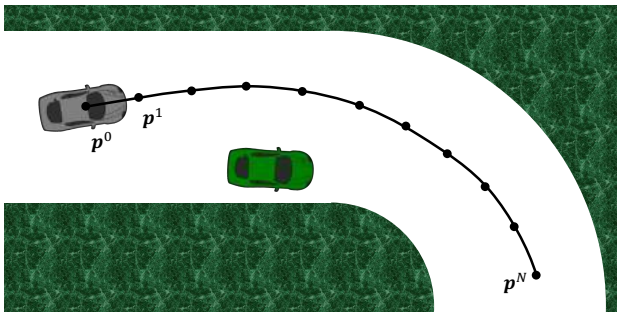


Fig. 2: Piecewise polynomial trajectory over an N-step planning horizon. Position of the nth waypoint is $\mathbf{p}^n$. Each trajectory section between two consecutive waypoints is represented using a second-order time polynomial.

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

$$\begin{aligned} x_n(t) &= \sum_{p=0}^{2} A_{n,p} t^p, \\ y_n(t) &= \sum_{p=0}^{2} B_{n,p} t^p,\ t \in [t_n, t_{n+1}] \end{aligned} \tag{11}$$

for $n = 0, 1, \ldots, N-1$. $\boldsymbol{A}_n = [A_{n,0}, A_{n,1}, A_{n,2}]$ and $\boldsymbol{B}_n = [B_{n,0}, B_{n,1}, B_{n,2}]$ are the coefficients for the nth polynomial in x and y direction, 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. 2.

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;

$$\begin{aligned}[x_n(t_n), y_n(t_n)] &= \mathbf{p}^n,\\ [x_n(t_{n+1}), y_n(t_{n+1})] &= \mathbf{p}^{n+1},\\ [\dot{x}_n(t_n), \dot{y}_n(t_n)] &= \boldsymbol{u}^n,\\ [\dot{x}_n(t_{n+1}), \dot{y}_n(t_{n+1})] &= \boldsymbol{u}^{n+1},\end{aligned} \tag{12}$$

for $n = 0, 1, \ldots, N-1$. Substitute (11) into the equations above, we obtain a set of equality constraints in $\boldsymbol{A}_n$ and $\boldsymbol{B}_n$.

The following constraints on speed, acceleration, and curvature should be enforced on each piece of the polynomial. For clarity of notations, we omit subscript n for the nth piece of polynomials, unless there is possible confusion.

2) Speed constraints: From (10), we have:

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

which is a second-order polynomial in t. Plug in (11), we obtain coefficients which are functions of $A_{:,:}$s and $B_{:,:}$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:

$$||\boldsymbol{u}^n|| \leq \bar{u}, \ \forall\, n, \tag{13}$$

which are quadratic constraints.

3) Acceleration constraints: For acceleration input, instead of using the nonlinear transform as shown in (10), we obtain a upper bound for it from bicycle model.

$$\begin{aligned}a^2(t) &= \ddot{x}^2 + \ddot{y}^2 - v^2(t)\dot{\theta}^2 &\text{(14a)}\\ &\leq \ddot{x}^2 + \ddot{y}^2. &\text{(14b)}\end{aligned}$$

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 non-negative. We neglect this term and relax the maximum acceleration constraint as:

$$\ddot{x}^2 + \ddot{y}^2 = A_{:,1}^2 + B_{:,1}^2 \leq \bar{a}^2, \tag{15}$$

where $\bar{a}$ represents maximum acceleration.

4) Curvature constraints: For bicycle model, steering angle is directly related to the geometry of the path and is completely decided by its 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 (10), 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:

$$\kappa_{\max} = \frac{2(A_{n,2}B_{n,1} - A_{n,1}B_{n,2})}{\min_{t\in[t_n,t_{n+1}]}\left[f(t)\right]^{\frac{3}{2}}}, \tag{16}$$

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

$$\kappa_{\max} \leq \bar{\kappa}_{\max} \tag{17}$$

using Sequential Quadratic Programming (SQP) [27].

C. Sensitivity Enhanced Iterated Best Response

To summarize, the optimization problem each player should solve is described in the following. For player i, the decision variables are: $\boldsymbol{A}_i \in \mathbb{R}^{3N}$, $\boldsymbol{B}_i \in \mathbb{R}^{3N}$ which are the coefficients 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 this approach is also known as collocation method [28]. For clarity, rewrite the optimization problem in the following form.

$$\begin{aligned}\max_{\boldsymbol{\theta}_i, \boldsymbol{A}_i, \boldsymbol{B}_i} \quad & s_i(\mathbf{p}_i^N) - s_j(\mathbf{p}_j^N) &\text{(18a)}\\ \text{s.t.} \quad & h_i(\boldsymbol{\theta}_i, \boldsymbol{A}_i, \boldsymbol{B}_i) = 0, &\text{(18b)}\\ & g_i(\boldsymbol{\theta}_i, \boldsymbol{A}_i, \boldsymbol{B}_i) \leq 0, &\text{(18c)}\\ & \gamma_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \leq 0, &\text{(18d)}\end{aligned}$$

where

- $h_i(\cdot)$ are the equality constraints involving only player i. This includes position and velocity continuity constraints (12).
- $g_i(\cdot)$ are the inequality constraints involving only player i. This include track constrains (2d), speed constrains (13), acceleration constraints (15), and curvature constrains (17).
- $\gamma_i(\cdot, \cdot)$ are the inequality constraints involving both players. This include the collision avoidance constraints as in (2c).

As in (5), we substitute the objective with the term that involves ego player and a sensitivity term that reflects the influence on opponent's optimal payoff. The only constraint that involves both players remains the same. 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.

$$\max_{\boldsymbol{\theta}_i, \boldsymbol{A}_i, \boldsymbol{B}_i} \quad s_i(\mathbf{p}_i^N) + \alpha \boldsymbol{\mu}_j \left.\frac{\partial \gamma_j}{\partial \boldsymbol{\theta}_i}\right|_{(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)} \boldsymbol{\theta}_i. \tag{19}$$

where $\boldsymbol{\theta}_i, \boldsymbol{A}_i, \boldsymbol{B}_i$ are subject to constraints (18b-18d). Note that α is a tunable parameter in the objective function. $\alpha = 0$ means the optimization function does not care about influencing the other player; while larger α leads to more aggressive behavior.

The algorithm is summarized in Algorithm 1. It is worth noting that the algorithm is solved by one player and always ends with the ego player's optimization problem.

V. Simulation Results

The performance of the proposed approach is first validated in simulated two-car competition scenarios. Note that although we use the kinematic bicycle model in the planner, our high-fidelity simulator employs full dynamics of the car with Fiala

Algorithm 1 Sensitivity Enhanced Iterated Best Response

L: maximum number of iterations in IBR
observe opponent's current states $\mathbf{p}^0, \boldsymbol{u}^0$
initialize opponent trajectory: $\boldsymbol{\theta}_j^1 = [\mathbf{p}_1, \ldots, \mathbf{p}_{N+1}]$
for $l = 1, 2, \ldots, L$ **do**
    ego: solve (19) using SQP with $\boldsymbol{\theta}_j = \boldsymbol{\theta}_j^l$
    obtain ego optimal strategy $\boldsymbol{\theta}_i^l$
    opponent: solve (19) using SQP with $\boldsymbol{\theta}_i = \boldsymbol{\theta}_i^l$
    obtain opponent optimal strategy $\boldsymbol{\theta}_j^l$
end for
ego: solve (19) with $\boldsymbol{\theta}_j = \boldsymbol{\theta}_j^L$

tire model [29], [30] and carefully calibrated system parameters of our experiment vehicle.

The proposed algorithm is implemented using Gurobi [1] solver with C++ interface. The non-convex constraints and objective function are imposed through linearization. Linearization is applied based on solution from the previous iteration. For the highly-nonlinear curvature constraints (17), we choose only to enforce the constraints on first half of the trajectory. We found 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 Robot Operating System (ROS) framework which handles message-passing.

The two players use the proposed game theoretic planner (GTP) and a naive Model Predictive Control (MPC) planner respectively. The naive MPC controller serves as a baseline strategy and only optimizes its own objective (4) using a initial guess of the opponent's behavior. In other words, the baseline planner does not exploit its influence on the other player.

We use the following parameters. The racing track is oval-shaped with total length $l_{\boldsymbol{\tau}} = 216\,\mathrm{m}$ and half-width $w_{\boldsymbol{\tau}} = 6.5\,\mathrm{m}$. The planning horizon is $5\,\mathrm{sec}$ and the planner updates at $2\,\mathrm{Hz}$. The maximum acceleration for both cars is $5\,\mathrm{m/s^2}$ and the maximum curvature is $0.11\,\mathrm{m^{-1}}$, 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 randomly selected positions, one always in front of the other. The maximum speed of the leading car is always set to a lower speed limit ($5\,\mathrm{m/s}$) than its competitor ($6\,\mathrm{m/s}$). The GTP car starts in the leading position in blocking tests and the secondary 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. 3. 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 could 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,

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

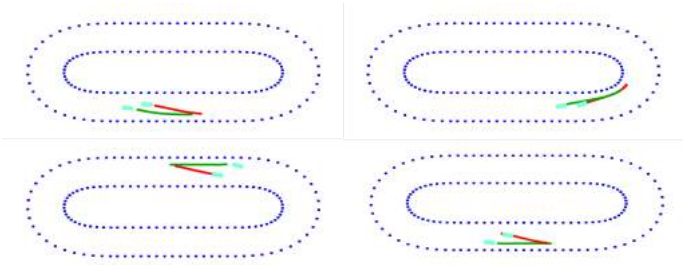
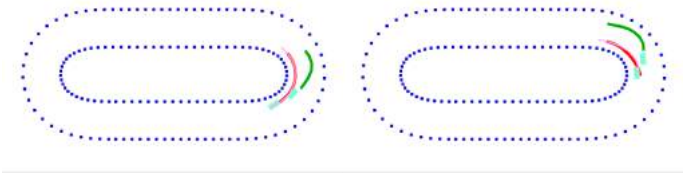
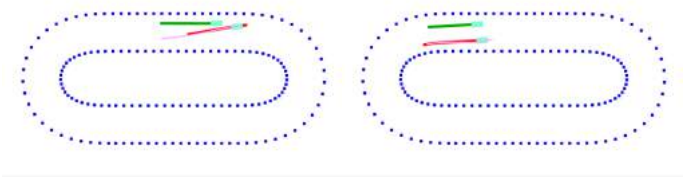


Fig. 3: Snapshots during a simulation in the blocking scenario. The GTP car (red trajectories) starts in the first position with lower speed limit with respect to the MPC car (green trajectories). The top right figure shows that the two cars entering a corner choose the short course. The other three 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.

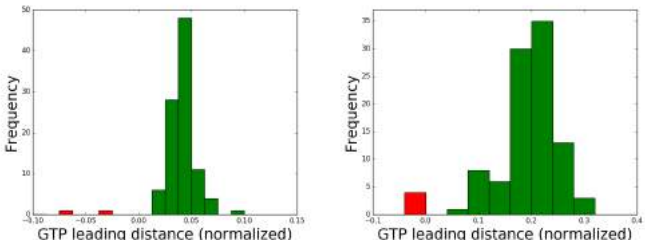


(a) Overtaking scenario 1.



(b) Overtaking scenario 2.

Fig. 4: Snapshots during simulations in the overtaking scenario. The GTP car (red trajectories) starts in the secondary position with a higher speed limit behind the MPC car (green trajectories). Only key snapshots during overtaking are shown here. In 4a, the MPC car passively avoids collision and steers away from its optimal trajectory. In 4b, the GTP car overtakes the MPC car from the left.



(a) Blocking scenario (b) Overtaking scenario

Fig. 5: 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.

they mostly choose the short course as optimal.

2) In the overtaking scenario, the GTP car uses the higher speed limit and starts slightly behind the MPC car. Note that this is exactly the same configuration as in blocking scenarios except that we flip the planning strategies of the two cars. Snapshots during simulations are shown in Fig. 4. Since the MPC car conservatively avoids collision with its opponent, we could observe that it may steer out of its optimal path and give way to the other car (Fig. 4a). In Fig.4b, 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 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. 5. The bins are colored green if the GTP car finishes the race first and red otherwise. We can see from Fig. 5a 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. 5b, 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.

VI. Experimental Results

A. RC car experiments

We first conducted experiments using scale RC car platform with Optitrack motion capture system. 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

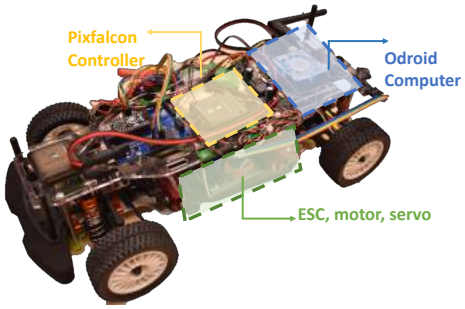
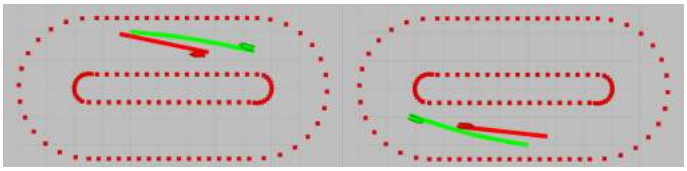
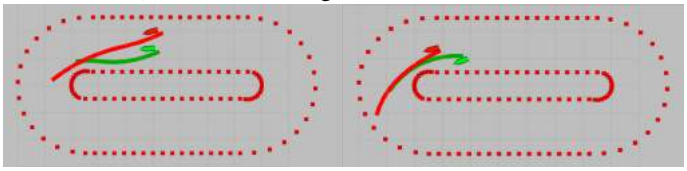


Fig. 6: 1/10 scale RC car experimental platform



(a) RC car blocking scenario visualization



(b) RC car overtaking scenario visualization

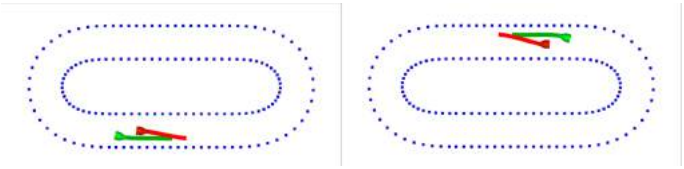
Fig. 7: Snapshots during RC car experiments. The red trajectories are for the GTP car and the green trajectories are for the MPC car. In 7a, the blocking scenario is shown where the GTP car adopts a trajectory in front of its opponent instead of going straight. In 7b, the GTP car aggressively drives pass the MPC car to overtake, knowing that the MPC car would slow down to avoid a collision.

structure of a RC car is shown in Fig. 6. The physical dimension of the cars are $20\,\mathrm{cm}$ wide and $40\,\mathrm{cm}$ long. An Optitrack motion capture system broadcasts pose information of both cars at $100\,\mathrm{Hz}$, 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, 2.7GHz Intel i7-7500U processor. The laptop runs two planner 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 $6\,\mathrm{m}$ and curved sections with radius $1.5\,\mathrm{m}$. Track half-width is $0.5\,\mathrm{m}$. The high and low speed limits are $2.5\,\mathrm{m/s}$ and $1.8\,\mathrm{m/s}$. The planning horizon is $2\,\mathrm{sec}$ and the planners update at $2\,\mathrm{Hz}$. The relative position at the start of the race is set to be the same as in simulations. We did 10 experiments of blocking and the GTP car won the races in all tests. We also conducted 10 experiments of overtaking, out of which 9 were successful. Each experiment consists of 2 laps in the blocking scenario or 1 successful overtaking. Snapshots of blocking and overtaking experiments are given in Fig. 7.

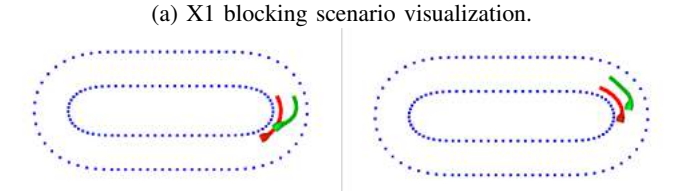
B. Full-sized car experiments

We also conducted experiments on X1, a student-built research vehicle. ¡ X1 is a student-built research vehicle as shown in Fig. 1b. X1 is a 4-wheel steer-by-wire, brake-by-wire, electrically driven vehicle. It is equipped with differential GPS and INS for precise position measurements. A low level RTP (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 [31] 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 game theoretic planner 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 game theoretic planner and the simulated car is using a MPC planner.

Snapshots plotted using recorded GPS data during the experiments are shown in Fig. 8. In the blocking scenario (Fig. 8a), we did 10 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 (Fig. 8b), we did 7 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.



(a) X1 blocking scenario visualization.



(b) X1 overtaking scenario visualization.

Fig. 8: Snapshots of X1 vs. a simulated car experiments. As before, the red trajectories are for the GTP car and the green trajectories are for the MPC car. 8a shows blocking behavior by the proposed planner and 8b shows overtaking while the MPC car gives way.

VII. Conclusion and Future Work

We present an online game-theoretic trajectory planner based on the concept of Nash equilibria. A modified Sensitivity Enhanced Iterated Best Response (SE-IBR) algorithm is used to solve for the approximate Nash equilibrium in the space of feasible trajectories for car-like robot kinematics. We leverage 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 is shown to be suitable for online planning and exhibits complex competitive behaviors in racing scenarios. The performance of the proposed planner is demonstrated in a two car racing game in both simulation and experiments with different vehicle hardware platforms with different scales.

In the future, we plan to benchmark our planner with human participants and study new methods to characterize the interaction with human drivers. Moreover, we plan to incorporate the full dynamics of the vehicle including the tire model in addition to kinematics to enable precise modeling and planning for highly dynamic and aggressive driving situations.

References

[1] 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," in *Proceedings of Robotics: Science and Systems*, June 2018.

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

[3] Z. Wang, T. Taubner, and M. Schwager, "Game theoretic planning for competitive real-time multi-drone racing in 3d environments," *Robotics and Autonomous Systems*, Under Review.

[4] 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)*, May 2018.

[5] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Proceedings of Robotics: Science and Systems*, June 2016.

[6] G. Ding, S. Aghli, C. Heckman, and L. Chen, "Game-theoretic cooperative lane changing using data-driven models," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2018.

[7] J. H. Yoo and R. Langari, "A stackelberg game theoretic driver model for merging," in *Proceedings of the ASME 2013 Dynamic Systems and Control Conference*, October 2013.

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

[9] 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 Transactions on Control Systems Technology*, vol. 26, no. 5, pp. 1782–1797, September 2018.

[10] 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. Vehicular Technology*, vol. 65, no. 6, pp. 3981–3992, 2016.

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

[12] A. Liniger and J. Lygeros, "A noncooperative game approach to autonomous racing," *IEEE Transactions on Control Systems Technology*, 2019.

[13] A. Turnwald, D. Althoff, D. Wollherr, and M. Buss, "Understanding human avoidance behavior: interaction-aware decision making based on game theory," *International Journal of Social Robotics*, vol. 8, no. 2, pp. 331–351, 2016.

[14] Z. Zhou, R. Takei, H. Huang, and C. J. Tomlin, "A general, open-loop formulation for reach-avoid games," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 6501–6506.

[15] J. S. Jang and C. Tomlin, "Control strategies in multi-player pursuit and evasion game," in *AIAA Guidance, Navigation, and Control*, August 2005, p. 6239.

[16] 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 *American Control Conference*, vol. 6, June 2003, pp. 4613–4619.

[17] S.-Y. Liu, Z. Zhou, C. Tomlin, and J. K. Hedrick, "Evasion of a team of dubins vehicles from a hidden pursuer," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 6771–6776.

[18] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[19] M. Wang, Z. Wang, S. Paudel, and M. Schwager, "Safe distributed lane change maneuvers for multiple autonomous vehicles using buffered input cells," in *2018 International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4678–4684.

[20] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 6271–6278.

[21] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes, "Collision avoidance and stabilization for autonomous vehicles in emergency scenarios," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1204–1216, July 2017.

[22] 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 *Int. Symp. on Experimental Robotics*, 2018.

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

[24] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

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

[26] R. M. Murray, M. Rathinam, and W. Sluis, "Differential flatness of mechanical control systems: A catalog of prototype systems," in *ASME international mechanical engineering congress and exposition*. Citeseer, 1995.

[27] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, p. 151, 1995.

[28] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[29] H. Pacejka, *Tire and vehicle dynamics*. Elsevier, 2005.

[30] H. B. Pacejka, *Tire Characteristics and Vehicle Handling and Stability*. Butterworth-Heinemann, December 2012, pp. 1–58.

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