Game Theoretic Motion Planning for Multi-Robot Racing

Zijian Wang[1], Riccardo Spica[1], and Mac Schwager[1]

Department of Aeronautics and Astronautics,
Stanford University, Stanford CA 94305, USA,
zjwang@stanford.edu

Abstract. This paper presents a real-time game theoretic planning algorithm for a robotic vehicle (e.g. a drone or a car) to race competitively against multiple opponents on a racecourse. Our algorithm plans receding horizon trajectories to maximally advance the robot along the racecourse, while taking into account the opponents' intentions and responses. We build on our previous work [5], which only considered racing with two robots. Our algorithm uses an iterative best response scheme with a new sensitivity term to find approximate Nash equilibria in the space of the multiple robots' trajectories. The sensitivity term seeks Nash equilibria that are advantageous to the ego robot. We demonstrate our approach through extensive multi-player racing simulations, where our planner exhibits rich behaviors such as blocking, overtaking, nudging or threatening, similar to what we observe from racing with human participants. Statistics also reveal that our game theoretic planner largely outperforms a baseline model predictive controller that does not consider the opponents' responses. Experiments are conducted with four quadrotor aerial robots to validate our approach in real time and with physical robot hardware.

Keywords: Game Theoretic Planning, Nash Equilibrium, Multi-Robot Systems

1 Introduction

Interactive behaviors among multiple agents without central coordination are complicated because the decision of one agent both depends on and also affects the decisions of other agents. While humans are good at reasoning about this interlaced dependency, for example, as pedestrian or driver on the road, this rich phenomenon is seldom captured in the existing robotics literature.

In this work, we address this challenging problem using a game theoretic planner that computes an approximate Nash equilibrium in the space of trajectories for an arbitrary number of robots. This algorithm builds upon our recent results for racing with only two robots [5]. The multi-robot case requires nontrivial algorithmic and theoretical advances beyond the two player game. We develop and test our algorithm in a multi-robot racing scenario, where the robot's objective is to finish the race along a bounded track ahead of other robots and without collision. Using our planner, the ego robot can explicitly account for the intention of other robots while planning its own action, using an iterative best response algorithm. The ego robot sequentially and repetitively solves

the optimization problem for itself and the opponents, based on the best strategy profiles of all the robots computed from the previous iteration. Crucially, the collision avoidance constraint is captured in a new sensitivity term in the planning iterations, which allow for the ego robot to plan to "nudge" the other robots out of its way to gain an advantage. We prove that if the iteration converges, then the solution satisfies the necessary conditions for a Nash equilibrium. We validate our algorithm with up to six robots in extensive simulation studies, in which the robots exhibit various competitive strategies, such as blocking, nudging, threatening, probing, similar to what human racers typically do in such competitions. We also use four quadrotor aerial robots to test our algorithm in real time under real-world disturbances and noises. The experimental results match our observation in simulation and successfully verify the applicability of our approach in real-life scenarios.

The solution given by our game theoretic planning resonates with numerous real-life scenarios involving multiple human agents. The approximate Nash equilibrium computed by our proposed algorithm naturally reflects the outcome of the competitive decision making process among multiple parties with conflicting interests. For example, during an interactive lane changing maneuver in the driving context, it is rare for the participants to either forcibly cut into the target lane regardless of the potential danger, or to immediately yield to the other car while sacrificing its own right-of-way. Instead, drivers evaluate tradeoffs constantly throughout the interaction, and progress is often made gradually. Because all the participants are competing, when they reach equilibrium, no one is at significant disadvantage. We find similar behaviors in our simulation studies. We argue that our algorithm provides a useful tool for motion planning in a non-cooperative multi-agent decision making context, and also provides a promising approach for human-robot interaction.

1.1 Related Work

Our work is related to a range of literature that applies game theory in estimation, planning and control for robots [7][8][9]. The multi-vehicle collision avoidance problem is formulated as a multi-agent differential game in [6] that enjoys provable safety guarantees. A two-player game between one human driver and one robot car is formulated in [1], where the robot accounts for the human's response using a learned cost function of the human driver through inverse reinforcement learning. A racing problem with two autonomous cars is investigated in [2], where the Nash and Stackelberg equilibrium can be found through a bimatrix game formulation given a finite set of pre-defined actions. Adversarial training is used in imitation learning [19] for controlling self-driving cars in a multi-agent setting. Our work goes beyond existing literature by planning in continuous trajectory space for multiple competing agents, and by seeking Nash equilibria, which are less conservative than a Stackelberg solutions.

Our work is also relevant to collision avoidance approaches in robotic navigation problems. The Optimal Reciprocal Collision Avoidance (ORCA) algorithm, a widely used benchmark method for collision avoidance, is proposed in [10] under a cooperative setting. The barrier method [4] gives a minimally invasive way to modify a nominal controller to resolve collision conflicts. Recently, the buffered Voronoi cell (BVC) [3][11] was proposed for communication-free reactive collision avoidance using only position

measurements. A centralized multi-robot trajectory generation algorithm is proposed in [13] that features fast computation and a safety guarantee. Intention inference of other interactive agents is useful for multi-agent motion planning. In [12], intention is modeled as multi-modal probabilistic distributions and learned through a conditional variational autoencoders (CVAEs). Bayesian inference can be used to predict the motion of nearby human [18] or robot [17], which is then used for subsequent safe motion planning. Many of these methods provide collision avoidance guarantees, but are not relevant to competitive scenarios among multiple robots.

2 Problem Formulation

Consider R robots in an adversarial racing game. For robot i, $i \in \{1, \cdots, R\}$, we represent its position as $p_i \in \mathbb{R}^2$. We assume single integrator dynamics, so that $\dot{p}_i = v_i$, where $v \in \mathbb{R}^2$ is the velocity of the robot. More complex dynamics can be accommodated in our algorithm with higher computational cost, however a single integrator model is adequate for many robotic platforms with existing low-level controllers, such as quadrotors and cars. The robot has a limit on its maximum speed, denoted as $\bar{v}_i > 0$. All agents have to obey the shared collision avoidance constraint, $\left\| p_j - p_i \right\| \geq \bar{d}_{ij}$, $\forall j \neq i$, $j \in \{1, \cdots, R\}$, where $\bar{d}_{ij} > 0$ is the minimum clearance between robot i and robot j, so that the robots do not collide with each other. No communication or central coordination is allowed among the robots, nevertheless, we assume that the robots can sense the position and velocity of its neighboring agents using onboard sensors. It is also assumed that the robots know the objectives, kinematic and dynamics constraints of the opponents so that they can apply game theoretic planning that involves inferences on the opponents' strategies. This is generally true for racing scenarios where all robots' objectives are to win the game and the robots' maximal velocities/accelerations are known.

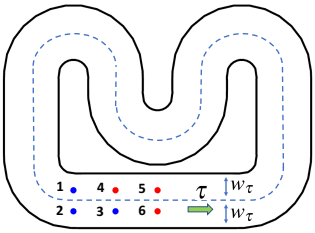


Fig. 1. Six robots in a bounded race track. The green arrow denotes the direction of the track. This setup is also used in the simulation studies in Section 4. The blue and red dots denote the nominal initial positions of the robots. To make the racing more interesting, the blue robots have higher maximal velocity than the red robots.

The robots race in a bounded track, represented by a center line curve τ and track width w_τ. See Figure 1 for an illustration. The center line is parameterized by a twice differentiable curve

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

that maps length to a coordinate in $\mathbb{R}^2$, where l_τ is the total length of the track. We can also find the track's unit normal vector, $n = \tau''/\|\tau''\|$, that points along the direction of the second-order derivative of the track curve. The robot needs to stay in the interior of the track at all times, i.e.,

$$\left\| n(s_i)^T [p_i - \tau(s_i)] \right\| \leq w_\tau,$$

where w_τ is the width of the track, and s_i is the point on the track center line that is closest to the robot's position p_i, defined as

$$s_i(p_i) = \arg\min_s \frac{1}{2} \|\tau(s) - p_i\|^2 .$$

The objectives of the robots are to maximize their own s_i (i.e. their own progress along the track) subject to the above-mentioned constraints. In this work, we aim to find the Nash equilibrium of the robots' strategy profiles. Unlike the case of single robot planning where an optimal solution often exists, there is no single optimal solution that can maximize the objective functions for a group of competing robots simultaneously. Instead, it is known that a Nash equilibrium is a reasonable outcome of such adversarial games since no robot can unilaterally improve its outcome. Formally, denote the strategy profile of robot i as θ_i, and the strategy of all other robots except i as $\theta_{-i} = \left\{ \theta_j | \forall j \neq i,\ j \in \{1, \cdots, R\} \right\}$. Also denote the payoff function that the robot tries to maximize as f_i. Then the strategy profile $(\theta_1^*, \theta_2^*, \cdots, \theta_R^*)$ is a Nash equilibrium if and only if

$$f_i(\theta_i^*, \theta_{-i}^*) \geq f_i(\theta_i, \theta_{-i}^*),\ \forall i \in \{1, \cdots, R\}. \tag{1}$$

In general, there are often many Nash equilibria (even continua of Nash equilibria) and finding a Nash equilibrium can be computationally prohibitive. In the next section, we will instead present our algorithm to find *approximate* Nash equilibrium in real time.

3 Multi-Robot Game Theoretic Planning

In this section, we first present our online algorithm for computing the approximate Nash equilibrium using an iterative best response approach with a sensitivity term. We then analyze the convergence property of the proposed algorithm and show that the resulting solution satisfies the necessary condition of the Nash equilibrium.

3.1 Iterative Best Response Algorithm

From the perspective of a single robot i, the racing game can be formulated as the following optimization problem in a receding horizon fashion, where the robot tries to maximize its own progress (2a) along the track, subject to dynamics (2b), collision

avoidance (2c), stay-within-track (2d) and input constraints (2e),

$$\max_{\theta_i} \; s_i(p_i^N) \tag{2a}$$

$$\text{s.t.} \quad p_i^k = p_i^{k-1} + u_i^k, \tag{2b}$$

$$\left\| p_j^k - p_i^k \right\| \geq \overline{d}_{ij}, \; \forall j \neq i, \; j \in \{1, \cdots, R\}, \tag{2c}$$

$$\left| n(p_i^k)^T [p_i^k - \tau(p_i^k)] \right| \leq w_\tau, \tag{2d}$$

$$\left\| u_i^k \right\| \leq \overline{u}_i, \tag{2e}$$

$$\forall k \in \{1, 2, \cdots, N\},$$

where we discretize the problem with step time dt and N is the horizon length that we consider. The input in (2b) therefore becomes the delta displacement $u_i^k = v_i dt$. The optimization above can be put into the following general form for brevity,

$$\max_{\theta_i} \; s_i(\theta_i) \tag{3a}$$

$$\text{s.t. } h_i(\theta_i) = 0, \tag{3b}$$

$$g_i(\theta_i) \leq 0, \tag{3c}$$

$$\gamma_{ij}(\theta_i, \theta_j) \leq 0, \; \forall j \neq i, \; j \in \{1, \cdots, R\}, \tag{3d}$$

where θ_i denotes the decision variable (strategy profile) composed of p_i^k and u_i^k. The equality constraint $h_i(\theta_i)$ corresponds to (2b). The inequality constraint $g_i(\theta_i)$ corresponds (2d) and (2e) and only depends on robot i's own strategy. The shared inequality constraint $\gamma_{ij}(\theta_i, \theta_j)$ corresponds to the collision avoidance constraint (2c) and depends jointly on robot i's and all its opponents' strategies. Note that the only interaction between multiple players is through the collision avoidance constraint (3d). Without this constraint, each robot would simply solve its own decoupled optimization problem.

Due to (3d), the optimization (3) is not directly solvable by robot i since the opponents' strategies θ_{-i} are unknown to robot i. To address this challenge, we propose an iterative best response (IBR) algorithm to compute the approximate Nash equilibrium. Starting from initial guesses on robots' strategies, IBR cycles through each robot to solve and update the optimal solution for that robot by fixing the strategies of other robots from the previous iteration, thereby relaxing the requirement of knowing the opponents' strategies in (3d). To retain the influence of the shared constraint (3d) on the objective function in the original problem (3), we need to augment the objective function in IBR with a term representing the sensitivity of robot j's strategy to robot i's strategy

$$\max_{\theta_i} \; s_i(\theta_i) - \sum_{j \neq i}^{R} \alpha_{ij} s_j^*(\theta_i), \tag{4}$$

where $s_j^*(\theta_i)$ is the best response of robot j given robot i's strategy, and $\alpha_{ij} > 0$ is a design parameter chosen by robot i with respect to robot j. We will prove in Theorem 1 that doing so allows the IBR solution to satisfy the necessary condition of Nash equilibrium associated with the original (not directly solvable) problem (3). Details on how

to compute $s_j^*(\theta_i)$ will be given later in Lemma 1. An intuitive explanation of (4) is that although robot j does not directly change robot i's position and objective function, indirect influence can be applied through the shared collision avoidance constraint, for instance, by blocking or nudging. The IBR iterations are repeated until the solution converges. In practice, we do not always wait for the convergence and we terminate the algorithm after a fixed number of iterations due to the real-time constraint. A quantitative study of the convergence is provided in Section 4.

As an illustrative example of how the IBR algorithm is executed, suppose that we are in the l-th iteration, given $\theta_1^{l-1}, \cdots, \theta_R^{l-1}$ from the previous iteration or from initialization. Then we sequentially obtain θ_i^l for $i = 1, \cdots, R$ by solving problem (4) under constraints (3b)-(3d), while keeping all other strategies, $\theta_j^l,\ j < i$ or $\theta_j^{l-1},\ j > i$ constant.

Now we present how to compute $s_j^*(\theta_i)$ in (4) without actually solving (3) for j, which would otherwise cause indefinite recursive calls when solving (4). The key insight here is to employ sensitivity analysis and study the change of $s_j^*(\theta_i)$ with respect to robot i's new strategy θ_i, based on the solution of the previous game iteration.

Lemma 1. *If $s_j^*(\theta_i)$ is the optimal value of an optimization problem (3) for robot j at the new game iteration where we are optimizing θ_i, then*

$$\left.\frac{ds_j^*}{d\theta_i}\right|_{\theta_i^l} = -\mu_{ji}^l \left.\frac{\partial \gamma_{ji}}{\partial \theta_i}\right|_{(\theta_i^l, \theta_j^l)}, \ \forall j \neq i \tag{5}$$

where θ_i^l and θ_j^l denote the latest best response of robot i and j already computed from the previous iteration. μ_{ji}^l is the Lagrange multiplier associated to the collision avoidance constraint of robot j with respect to robot i, i.e., γ_{ji}, when computing θ_j^l in the previous iteration.

Proof. Consider solving (3) for robot j at a new game iteration given a new θ_i, and we are interested in finding $\frac{\mathrm{d}s_j^*}{\mathrm{d}\theta_i}$. Suppose that the Lagrange multipliers associated with constraints (3b), (3c), (3d) when solving for θ_j^l in the previous iteration are λ_j^l, ν_j^l, μ_{ji}^l respectively. Then utilizing standard local sensitivity arguments [14],

$$\frac{\mathrm{d}s_j^*}{\mathrm{d}\theta_i} = -\lambda_j^l \frac{\mathrm{d}h_j(\theta_j^l)}{\mathrm{d}\theta_i} - \nu_j^l \frac{\mathrm{d}g_j(\theta_j^l)}{\mathrm{d}\theta_i} - \sum_{k \neq i,j}^{R} \mu_{jk}^l \frac{\partial \gamma_{jk}(\theta_j^l, \theta_k^l)}{\partial \theta_i} - \mu_{ji}^l \frac{\partial \gamma_{ji}(\theta_j^l, \theta_i^l)}{\partial \theta_i} \tag{6}$$

$$= -\mu_{ji}^l \frac{\partial \gamma_{ji}(\theta_j^l, \theta_i^l)}{\partial \theta_i}, \tag{7}$$

where the first three terms in (6) are all zero since those constraints do not change with respect to θ_i. This completes the proof. □

Utilizing Lemma 1, we can write down the first-order approximation of $s_j^*(\theta_i)$ as

$$s_j^*(\theta_i) = s_j^*(\theta_i^l) + \left.\frac{\mathrm{d}s_j^*}{\mathrm{d}\theta_i}\right|_{\theta_i^l} (\theta_i - \theta_i^l). \tag{8}$$

Plugging (8) into (4), we can drop the terms that do not depend on the new strategy θ_i, i.e., $s_j^*(\theta_i^l)$ and θ_i^l, when carrying out the maximization. Then after incorporating the sensitivity analysis, the objective function (4) can be rewritten as

$$\max_{\theta_i} s_i(\theta_i) + \sum_{j \neq i}^{R} \alpha_{ij} \mu_{ji}^l \left. \frac{\partial \gamma_{ji}}{\partial \theta_i} \right|_{(\theta_i^l, \theta_j^l)} \theta_i, \tag{9}$$

which is what we actually use in Section 4. We also decay α_{ij} exponentially to zero in every iteration in our implementation as explained in Section 3.2.

3.2 Equilibrium Analysis

If the solution of the iterative best response algorithm converges, then it suggests that no robot can further improve its own objective function unilaterally and thus satisfies the definition of Nash equilibrium in (1). However, due to the modification in (4), we still need to show that a Nash equilibrium with the modified objective functions is also a Nash equilibrium in the problem's original form in (2). This proved in the following result.

Theorem 1. *If $\gamma_{ij}(\theta_i, \theta_j) = \gamma_{ji}(\theta_j, \theta_i)$ and the IBR algorithm converges to a solution $(\theta_1^l, \cdots, \theta_R^l)$, then the strategy tuple $(\theta_1^l, \cdots, \theta_R^l)$ satisfies the necessary conditions for a Nash equilibrium.*

Proof. Apply KKT conditions for (3), we obtain

$$\frac{\partial s_i}{\partial \theta_i}(\theta_i^*) - \sum_{j \neq i}^{R} \mu_{ij}^* \frac{\partial \gamma_{ij}}{\partial \theta_i}(\theta_i^*, \theta_j^*) - \lambda_i^* \frac{\partial h_i}{\partial \theta_i}(\theta_i^*) - \nu_i^* \frac{\partial g_i}{\partial \theta_i}(\theta_i^*) = 0 \tag{10a}$$

$$h_i(\theta_i^*) = 0 \tag{10b}$$

$$g_i(\theta_i^*) \leq 0 \tag{10c}$$

$$\nu_i^* g_i(\theta_i^*) = 0, \nu_i^* \geq 0 \tag{10d}$$

$$\gamma_{ij}(\theta_i^*, \theta_j^*) \leq 0, \ \forall j \neq i, \ j \in \{1, \cdots, R\} \tag{10e}$$

$$\mu_{ij}^* \gamma_{ij}(\theta_i^*, \theta_j^*) = 0, \mu_{ij}^* \geq 0, \ \forall j \neq i, \ j \in \{1, \cdots, R\} \tag{10f}$$

Apply KKT conditions to (9), we have

$$\begin{aligned} &\frac{\partial s_i}{\partial \theta_i}(\theta_i^l) + \sum_{j \neq i}^{R} \alpha_{ij} \mu_{ji}^l \frac{\partial \gamma_{ji}}{\partial \theta_i}(\theta_i^l, \theta_j^l) - \sum_{j \neq i}^{R} \mu_{ij}^l \frac{\partial \gamma_{ij}}{\partial \theta_i}(\theta_i^l, \theta_j^l) \\ &\qquad - \lambda_i^l \frac{\partial h_i}{\partial \theta_i}(\theta_i^l) - \nu_i^l \frac{\partial g_i}{\partial \theta_i}(\theta_i^l) = 0 \end{aligned} \tag{11a}$$

$$h_i(\theta_i^l) = 0 \tag{11b}$$

$$g_i(\theta_i^l) \leq 0 \tag{11c}$$

$$\nu_i^l g_i(\theta_i^l) = 0, \nu_i^l \geq 0 \tag{11d}$$

$$\gamma_{ij}(\theta_i^l, \theta_j^l) \leq 0, \ \forall j \neq i, j \in \{1, \cdots, R\} \tag{11e}$$

$$\mu_{ij}^l \gamma_{ij}(\theta_i^l, \theta_j^l) = 0, \mu_{ij}^l \geq 0, \ \forall j \neq i, j \in \{1, \cdots, R\} \tag{11f}$$

The objective here is to show that the solution to the KKT conditions (11) can also satisfy the KKT conditions (10). Equations (10a)-(10e), and (11a)-(11e) are equivalent if

$$\lambda_i^* = \lambda_i^l, \;\; \nu_i^* = \nu_i^l,$$

$$\mu_{ij}^* = \mu_{ij}^l - \alpha_{ij}\mu_{ji}^l, \; \forall j \neq i, \; i,j \in \{1,\cdots,R\}.$$

To ensure (10f) is satisfied, we need to impose that

$$(\mu_{ij}^l - \alpha_{ij}\mu_{ji}^l)\gamma_{ij}(\theta_i^l, \theta_j^l) = 0 \tag{12a}$$

$$\mu_{ij}^l \geq \alpha_{ij}\mu_{ji}^l \tag{12b}$$

$$(\mu_{ji}^l - \alpha_{ji}\mu_{ij}^l)\gamma_{ji}(\theta_j^l, \theta_i^l) = 0 \tag{12c}$$

$$\mu_{ji}^l \geq \alpha_{ji}\mu_{ij}^l. \tag{12d}$$

where (12a) and (12c) are already satisfied by (11f), and the fact that $\gamma_{ij}(\theta_i^l, \theta_j^l) = \gamma_{ji}(\theta_j^l, \theta_i^l)$. For (12b) and (12d) however, we need to choose sufficiently small α_{ij} such that

$$\mu_{ij}^l \geq \alpha_{ij}\mu_{ji}^l, \; \forall j \neq i, \; i,j \in \{1,\cdots,R\}.$$

The choice of α_{ij} is made possible in practice by exponentially decaying α_{ij} to zero in the iterative best response optimization. □

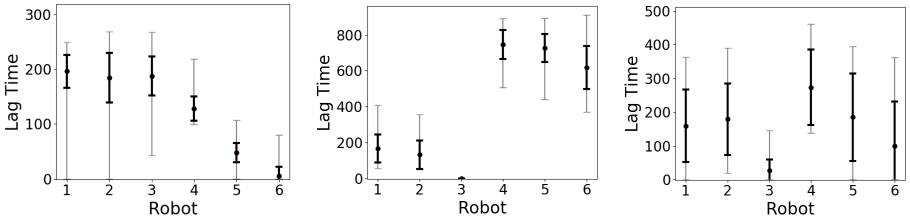
Remark 1. The procedure of decaying α_{ij} exponentially to zero is analogous to shrinking the step size in the gradient descent for solving general optimization problems.

4 Simulation

We conduct extensive simulation studies to validate the proposed algorithm and also shed some light on the behavior of the planner during different racing scenarios. For comparison, we run our game theoretic planner (GTP) against a model predictive controller (MPC), where the agent only plans for its own trajectory while naively assuming the opponents will travel in a straight line along the track. Note that such MPC approaches are the state of the art for autonomous car racing [15, 16]. We consider a racing game with 6 robots, equally divided into three faster robots (max velocity $0.6m/s$) and three slower robots (max velocity $0.5m/s$). The faster robots are initialized behind the slower robots along the track, as shown in Figure 1, to enforce interesting interactions in the game. Each robot has a radius $0.3m$ and it has to maintain a minimal clearance $0.8m$ with respect to all other robots. The game is fully competitive, meaning that each robot selfishly optimizes its own objective without cooperating with any other robot. All the robots plan their trajectories 3 seconds into the future with 10 planning steps. The maximal number of game iterations is 10 for the game theoretic planner to ensure online planning and execution. We implement our simulation in ROS and C++, with

Gurobi[1] as the optimization solver. Utilizing multiple computing cores on a Intel i7-6700HQ CPU, where the simulation is conducted, we run the planners in parallel on 6 different CPU cores.

We run 100 simulations for each of the three scenarios: (a) faster MPC vs slower GTP; (b) faster GTP vs slower MPC; (c) faster GTP and slower GTP. All robots' initial positions are randomly perturbed around their nominal values shown in Figure 1. Additional randomness also comes from the time delay due to the uncontrollable CPU task scheduling and the fact that the planners are running asynchronously. The robots are required to finish two laps. The statistical results of the simulations are shown in Figure 2, where robot 1–3 always mean the faster robots while robot 4–6 always mean the slower robots. The metric we choose for evaluating the performance is *lag time* (*y* axis in the plot), that is, the difference between an agent's finish time and the winner of the game. Consequently, *lag time* 0 means the first to finish a particular game.



(a) Faster MPC vs slower GTP (b) Faster GTP vs slower MPC (c) Faster GTP vs slower GTP

Fig. 2. Statistics of 100 simulations for each of the three scenarios. Robot index 1–3 means faster robots while 4–6 means slower robots. *y* axis indicates the *lag time*, i.e., the time difference behind the winner of the game. Black dots are the mean *lag time*. Standard deviation are shown by the bold black vertical bars, and min/max values are represented by the gray bars.

From Figure 2(a), GTP robots win most of the game despite being slower than the MPC robots, verifying the GTP's ability to block the opponents. The MPC robots do occasionally win the game due to the speed advantage, but overall have higher mean lag time. In comparison, faster GTP in Figure 2(b) outperforms MPC robot by a large margin. More interestingly, Figure 2(c) depicts the game with six GTP robots with different speeds. The result is as expected and the winners are more evenly distributed with smaller lag time differences compared to (a) and (b), presumably because all agents are exploiting Nash equilibrium to avoid being disadvantageous in this case. Some sample snapshots of the simulation process are provided in Figure 3, 4 and 5, where the colored lines denote the planned trajectories. A large bulk of simulation videos are available online: https://msl.stanford.edu/dars18-planning-video.

To verify and analyze the convergence of our planner in practice, we run scenario (c) with different number of maximal number of game iterations, and the results are reported in Figure 6. We also report the speed of the planner in Table 1. Note that one game iteration here means doing one round of planning for the ego robot and for

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

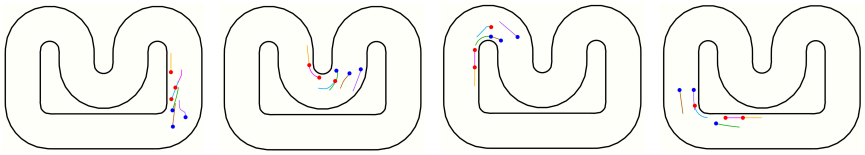


Fig. 3. Sample snapshots of scenario (a): faster MPC (blue) vs slower GTP (red) game. The GTP agents exhibit numerous blocking behaviors. After one lap, the slower GTP robots still maintain more advanced positions than the faster MPC robots.

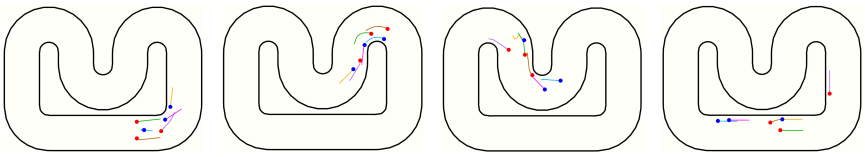


Fig. 4. Sample snapshots of scenario (b): faster GTP (red) vs slower MPC (blue) game. The faster GTP robots are able to quickly overtake the naive MPC robots. After one lap, the MPC robots are left behind and separated from the GTP robots by a large distance.

each opponent. As expected, more iterations result in better convergence, as shown in Figure 6. Within 10 game iterations, the planner is able to converge for most of the time. However, the tradeoff is computation speed, as shown in Table 1. In practice, we can use these results to properly tradeoff between accuracy and speed. In our experience, 2 game iterations perform similarly to 10 but much faster, in spite of not fully converging as shown in Figure 6. A video is provided for this comparison.

Table 1. Planner speed with six robots under different maximal number of game iterations.

Max number of game iterations	2	5	10
Planner frequency (Hz)	8.0	3.8	2.4

5 Experiments

Hardware experiments are conducted with four custom-built quadrotor aerial robots, as shown in Figure 7. We use the Pixfalcon open-source flight controller board as an integrated solution for all the flight electronics. The PX4 open-source autopilot software runs on the Pixfalcon hardware and manages low-level and real-time state estimation and attitude control. Each quadrotor is equipped with an Odroid XU4 single board computer, which is responsible for high-level planning and control (for example, tracking a given trajectory). The Odroid runs the Robot Operating System (ROS) in order to interface with the game planner and also send desired velocity commands to the Pixfalcon. The experiment was performed in a motion capture room (size 16.5m by 6.5m) with

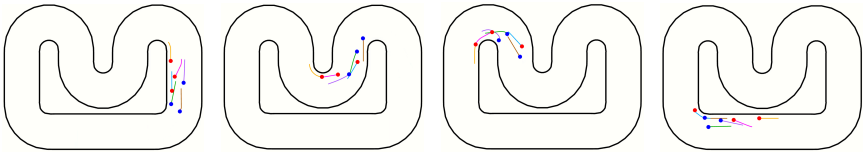
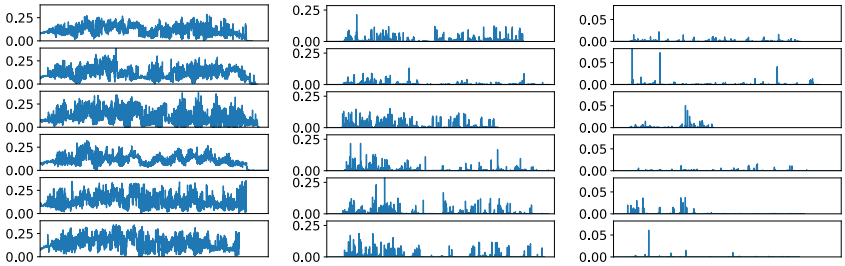


Fig. 5. Sample snapshots of scenario (c): faster GTP (blue) vs slower MPC (red) game. The game is more fierce compared to scenario (a) and (b) as the robots try to block and overtake at the same time. The robots also stick together more tightly throughout the game, indicating the dramatically increased difficulty in winning the game among multiple GTP robots.



(a) Max 2 game iterations (b) Max 5 game iterations (c) Max 10 game iterations

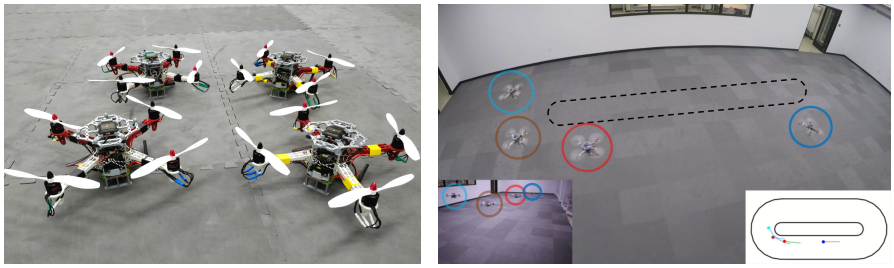
Fig. 6. Convergence visualization under different maximal number of game iterations over time in the scenario of fast GTP vs slower GTP. The plots show the mean position residue per waypoint in the planned trajectory between the last iteration and the second last iteration. A zero residue means that the solution has converged. From top to bottom: correspond to robot 1 to 6 respectively. These plots verify that our game planner does converge in practice given enough iterations, and also provide guidelines on how we can tradeoff between accuracy and speed.

the OptiTrack system. The position and heading measurement from the motion capture system are fed into EKF estimator on Pixfalcon for sensor fusion with the onboard IMU.

The robots operate at a constant altitude and race around an oval-shaped track. All four robots use the game theoretic planner. As in the simulation, two robots that start in the front have a slower maximal speed at $0.8m/s$, while the two in the back have higher maximal speed at $1.0m/s$. We set the minimal clearance between robots, $\overline{d}_{ij}$, to $0.6m$, slightly larger than the radius of the robot, $0.4m$. We use the time horizon $2s$ with 10 planning steps, resulting in $0.2s$ separation between two adjacent positions on the planned trajectory. We choose to use 2 game iterations in order to trade the optimality of the solution for speed considering the fast and real-time nature of the experiment. This leads to about $12Hz$ trajectory planning for four robots using the same computing device as in Section 4.

Experiment video is accessible at https://msl.stanford.edu/dars18-planning-video. The outcome is consistent with our observations in simulation. The robots exhibit very competitive behaviors. On the one hand, the slower robots were good at blocking their opponents, which are 25% faster. It was not until $200s$, or 8 laps, after the start of the

game that the first overtaking was done. On the other hand, the faster robots did overtake their slower counterparts, after aggressively seeking opportunities in various ways. The experiment shows that our proposed algorithm maintains its performance on physical robots and meets the real-time requirements of real-world scenarios. It is also worth noting that there was no collision during the experiment.



(a) Custom-built quadrotor robot

(b) Snapshot of the experiment. The planned trajectories are shown in the lower right corner

Fig. 7. Experiment with four quadrotor robots, all running game theoretic planner. Video can be watched online at https://msl.stanford.edu/dars18-planning-video

6 Conclusion and Future Work

In this paper, we propose a game theoretic planner for a multi-robot racing scenario, where the robots have competing objectives and interact through shared collision avoidance constraints. The game planner computes the approximate Nash equilibrium in real time using an iterative best response algorithm, in which the robots plan trajectories for themselves and their opponents in order to take informed decisions by considering the potential intention and response of the opponents. We prove that the solution satisfies the necessary condition of the Nash equilibrium (Theorem 1). Extensive simulation studies and a hardware experiment are presented to validate our approach. Statistics also verify that our game theoretic planner outperforms its non-game counterpart in all three benchmark racing scenarios.

There also exist many avenues for future improvement of this work. We intend to employ machine learning, specifically imitation learning, to incorporate more information about the opponents' behaviors through past observations. Another potential advantage of using learning techniques is that we can train a neural network offline to approximate the solution of the game planner, which can then be used as a better initial guess to reduce the number of iterations during the real-time optimization process to speed up the solution. Finally, we are also interested in extending our approach to 3D environments and more complex robot dynamics.

Acknowledgement This work was supported by the Toyota Research Institute (TRI). This article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. The authors are grateful for this support.

References

1. D. Sadigh, S. Sastry, S. Seshia, and A. Dragan. Planning for Autonomous Cars that Leverage Effects on Human Actions. In Robotics: Science and Systems (2016)
2. A. Liniger and John Lygeros. A Non-Cooperative Game Approach to Autonomous Racing. arXiv preprint arXiv:1712.03913 (2017)
3. 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)
4. L. Wang, A. Ames, and M. Egerstedt. Safety Barrier certificates for collisions-free multirobot systems. IEEE Transactions on Robotics, vol. 33, no. 3, pp. 661–674 (2017)
5. R. Spica, D. Falanga, E. Cristofalo, E. Montijano, D. Scaramuzza, and M. Schwager. A Game Theoretic Approach to Autonomous Two-Player Drone Racing. arXiv preprint arXiv:1801.02302 (2018)
6. M. Chen, J. Shih, and C. Tomlin. Multi-vehicle collision avoidance via hamilton-jacobi reachability and mixed integer programming. Decision and Control (CDC), IEEE 55th Conference on, pp. 1695–1700 (2016)
7. S. Nikolaidis, S. Nath, A. Procaccia and S. Srinivasa. Game-theoretic modeling of human adaptation in human-robot collaboration. Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, pp. 323–331 (2017)
8. R. Vidal, O. Shakernia, J. Kim, D. Shim and S. Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. IEEE transactions on robotics and automation, vol. 18, no. 5, pp. 662–669 (2002)
9. Z. Zhang, L. Zhou and P. Tokekar. Strategies to Design Signals to Spoof Kalman Filter. American Control Conference (ACC), pp. 5837–5842 (2018)
10. J. Van Den Berg, S. Guy, M. Lin and D. Manocha. Reciprocal n-body collision avoidance. In 14th International Symposium on Robotics Research, pp. 3–19 (2011)
11. M. Wang and Z. Wang and S. Paudel and M. Schwager. Safe Distributed Lane Change Maneuvers for Multiple Autonomous Vehicles Using Buffered Input Cells. International Conference on Robotics and Automation (ICRA), pp. 4678–4684 (2018)
12. E. Schmerling, K. Leung, W. Vollprecht and M Pavone. Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction. International Conference on Robotics and Automation (ICRA), pp. 3399–3406 (2018)
13. S. Tang, J. Thomas and V. Kumar. Hold Or take Optimal Plan (HOOP): A quadratic programming approach to multi-robot trajectory generation. The International Journal of Robotics Research, vol. 37, no. 9, pp. 1062–1084 (2018)
14. S. Boyd and L. Vandenberghe. Convex optimization. Cambridge university press (2004)
15. R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl. Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm. European Control Conference (ECC), pp. 141–147 (2016)
16. 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)
17. H. Nishimura and M. Schwager. Active Motion-Based Communication for Robots with Monocular Vision. International Conference on Robotics and Automation (ICRA), pp. 2948–2955 (2018)
18. Jaime F. Fisac, A. Bajcsy, S. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin, and A. D. Dragan. Probabilistically Safe Robot Planning with Confidence-Based Human Predictions. Robotics Science and Systems (RSS) (2018)
19. R. Bhattacharyya, D. P. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer. Multi-agent imitation learning for driving simulation. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018)