A Real-Time Game Theoretic Planner for Autonomous Two-Player Drone Racing

Riccardo Spica, Davide Falanga, Eric Cristofalo, Eduardo Montijano, Davide Scaramuzza, Mac Schwager

***Abstract*—To be successful in multi-player drone racing, a player must not only follow the race track in an optimal way, but also compete with other drones through strategic blocking, faking, and opportunistic passing while avoiding collisions. Since unveiling one's own strategy to the adversaries is not desirable, this requires each player to independently *predict* the other players' future actions. Nash equilibria are a powerful tool to model this and similar multi-agent coordination problems in which the absence of communication impedes full coordination between the agents. In this paper, we propose a novel receding horizon planning algorithm that, exploiting sensitivity analysis within an iterated best response computational scheme, can approximate Nash equilibria in real time. We demonstrate that our solution effectively competes against alternative strategies in a large number of drone racing simulations.**

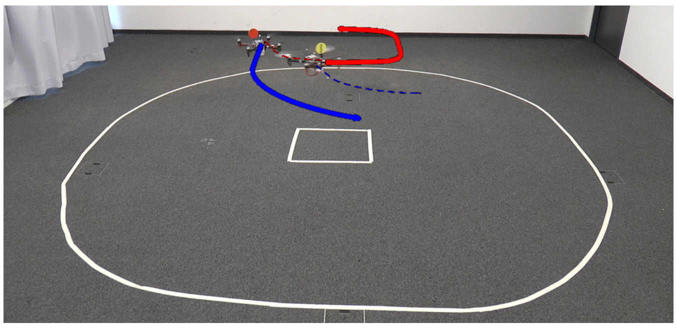


Figure 1: Drone racing experiment. A video of the experiments can be found at [16].

I. INTRODUCTION

Drone racing has recently become a popular sport with international competitions being held regularly and attracting a growing public [1]. In these races, human pilots directly control the UAVs through a radio transmitter while receiving a first-person-view live stream from an onboard camera. Human racers need years of training to master the advanced navigation and control skills that are required to be successful in this sport. Many of these skills would certainly prove useful for a robot to safely and quickly move through a cluttered environment in, for example, a disaster response scenario. For this reason, drone racing has attracted a significant interest from the scientific community, which led to the first autonomous drone racing competition being held during the IROS 2016 international conference [2].

Most of the past research has focused on a time trial style of racing: a single robot must complete a racing track in the shortest amount of time. This scenario poses a number of challenges in terms of dynamic modeling, on-board perception, localization and mapping, trajectory generation and optimal control. Impressive results have been obtained in this context not only for autonomous UAVs [3], but also for a variety of different platforms, such as cars [4]–[6] motorcycles [7], and even sailboats [8]. Much less attention, on the other hand, has been devoted to the more classical multi-player style of racing that we address in this paper. In addition to the aforementioned challenges, this kind of race also requires direct competition with other agents, incorporating strategic blocking, faking, and opportunistic passing while avoiding collisions. Multi-player drone racing is then also a good testing ground for developing and testing more widely applicable non-cooperative multi-robot planning strategies.

A number of single-agent planning strategies can effectively deal with both fixed and moving obstacles such as other racers. See, for example, [9]–[13], and [14], which considers the car racing application. Most of these works, however, rely on simple "open-loop" models to predict the obstacle motion. By not taking into account the *reactive* behavior of other players, these solutions can induce oscillatory effects sometimes referred to as *reciprocal dances* [15]. Moreover, by properly anticipating other players reactions, one can realize more sophisticated and effective strategies.

In this paper, we consider two-player autonomous drone racing as a practical application to develop effective strategies for scenarios involving multiple rational agents that: i) do not communicate their policies to each other, ii) know each other's goals and constraints, iii) behave reactively in order to avoid collisions with other agents. We believe that game theory [17] is the most appropriate tool to model scenarios such as these.

While broadly used in economics and social science, game theory has not yet attracted, in our opinion, a sufficient interest from the robotics community, mostly due to the computational complexity typically associated with these methods.

Some interesting results have been obtained in the context of robust control design where the disturbance acting on a system can be modeled as an antagonistic agent thus giving rise to a zero-sum differential game [18], [19]. More recently, in [20],

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. This research was also supported by ONR grant N00014-16-1-2787 and a National Defense Science and Engineering Graduate Fellowship. We are grateful for this support.

This research was supported by the SNSF-ERC Starting Grant and the National Centre of Competence in Research (NCCR) Robotics, through the Swiss National Science Foundation.

This research was supported by the Spanish project Ministerio de Economía y Competitividad DPI2015-69376-R.

R. Spica, E. Cristofalo and M. Schwager are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA `rspica;ecristof;schwager@stanford.edu`

D. Falanga and D. Scaramuzza are with the Robotics and Perception Group, Dep. of Informatics, University of Zurich, and Dep. of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland–`http://rpg.ifi.uzh.ch`.

E. Montijano is with Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, Zaragoza 50018, Spain `emonti@unizar.es`.

[21], the interaction between an autonomous car and a human driven one was modeled as a Stackelberg game: the human is assumed to know in advance what the autonomous car will do and to respond optimally according to an internal cost function. Giving the other players some information advantage can, in general, improve the robustness of the system but also result in overly conservative actions. A more realistic model for racing applications is that of Nash equilibria which, instead, assume a fully symmetric information pattern. A very recent paper [22] proposes a control algorithm for coordinating the motion of multiple cars through an intersection exploiting generalized Nash equilibria. In the context of car racing, the authors of [23] investigate both Stackelberg and Nash equilibria. In both these works, however, computational performance close to real time can only be obtained in simplified scenarios. The authors of [23] also discuss the importance of exploiting blocking behaviors in racing. However, while in our work such behaviors naturally emerge from the use of sensitivity analysis, in [23] these are hardcoded in the payers' cost function.

Motivated by the success obtained by Model Predictive Control (MPC) in the development of real-time optimal control schemes, we apply similar receding horizon control strategies in the context of multi-player drone racing. Differently from a standard MPC planner, however, our strategy also takes into account other agents reactions to the ego agent actions. We achieve this by employing an iterated best response computation scheme: each player alternatively solves an optimal control problem for *each* player while keeping the other player's strategy constant. In addition to this, in order to fully capture and exploit the effects of the collision avoidance constraints, we also use sensitivity analysis to approximate the effects of one player's actions on its opponent's cost.

Despite the fact that Nash equilibria are often difficult to achieve or verify in dynamic games, we prove that, if our algorithm converges, the output satisfies necessary conditions for a Nash equilibrium. In practice, we find that the algorithm does converge, providing a theoretical foundation for our technique. The algorithm also runs in real time, at 20Hz, on standard hardware. We demonstrate the effectiveness of our approach in a large number of simulations in which our planner competes against multiple alternative strategies.

The rest of the paper is organized as follows. First, in Sect. II, we model the drone racing problem and introduce the associated sensing and control constraints. Then, in Sect. III, we formulate the control problem as a Nash equilibrium search and we detail the numerical methods used to obtain real time solutions. In Sect. IV we report simulation results obtained by letting our method compete against alternative ones. Finally, in Sect. V, we conclude the paper and outline future extensions.

II. PRELIMINARIES

Consider two quadrotor UAVs competing against each other in a drone racing scenario. In order to simplify the high level control strategy, we will assume that the robots fly at a constant altitude with simplified holonomic discrete time dynamics given by

$$\boldsymbol{p}_i^k = \boldsymbol{p}_i^{k-1} + \boldsymbol{u}_i^k \tag{1}$$

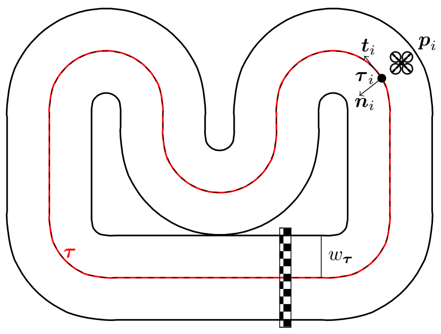


Figure 2: Representation of the race track used for the simulations. The track is parameterized by its center line $\boldsymbol{\tau}$ and its half width $w_{\boldsymbol{\tau}}$. Given the current robot position $\boldsymbol{p}_i$, we can define a local track frame with origin $\boldsymbol{\tau}_i$ as the closest point to $\boldsymbol{p}_i$ and with $\boldsymbol{t}_i$ and $\boldsymbol{n}_i$ being the local tangent and normal vectors to the track in $\boldsymbol{\tau}_i$.

where $\boldsymbol{p}_i \in \mathbb{R}^2$ is the robot horizontal position in the world frame, and $\boldsymbol{u}_i \in \mathbb{R}^2$ is the robot linear velocity, which we control directly as the input. We also assume that the players can measure their opponent's position by exploiting onboard cameras and visual markers attached to the robots frames.

Due to limitations of onboard actuators, the robots linear velocities are limited, i.e.

$$\|\boldsymbol{u}_i\| \leq \overline{u}_i \in \mathbb{R}^+.$$

The race track center line is defined by a twice continuously differentiable immersed plane closed curve $\boldsymbol{\tau}$ (see Fig. 2). For such a curve, there exists an arch-length parameterization

$$\boldsymbol{\tau} : [0, l_{\boldsymbol{\tau}}] \mapsto \mathbb{R}^2, \text{ with } \boldsymbol{\tau}(0) = \boldsymbol{\tau}(l_{\boldsymbol{\tau}})$$

where $l_{\boldsymbol{\tau}}$ is the total length of the track. Moreover, one can also define a local signed curvature κ and unit tangent and normal vectors ($\boldsymbol{t}$ and $\boldsymbol{n}$ respectively) as follows

$$\boldsymbol{t} = \boldsymbol{\tau}' \tag{2}$$

$$\boldsymbol{n}\kappa = \boldsymbol{\tau}''. \tag{3}$$

To remain within the boundaries of the track, the robot's distance from the track center line must be smaller than the (constant) track width $w_{\boldsymbol{\tau}} \in \mathbb{R}^+$, i.e.

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

where $s_i \in [0, l_{\boldsymbol{\tau}}]$ is the robot position along the track, i.e. the arch length of the point on the track that is closest to $\boldsymbol{p}_i$

$$s_i(\boldsymbol{p}_i) = \arg\min_s \tfrac{1}{2} \|\boldsymbol{\tau}(s) - \boldsymbol{p}_i\|^2. \tag{4}$$

In order to avoid potential collisions, each robot always maintains a minimum distance $\overline{d}_i \in \mathbb{R}^+$ with respect to its opponent, i.e.

$$\left\|\boldsymbol{p}_i - \boldsymbol{p}_j\right\| \geq \overline{d}_i. \tag{5}$$

Note that here, as well as in the rest of the paper, we always use i ($= 1$ or 2) to refer to a generic robot and j ($= 2$ or 1 respectively) to refer to its opponent.

Since we exploit a receding horizon control approach, the objective for each robot is to have a more advanced position along the track, with respect to the opponent, at the end of the planning horizon N. The final position is given by:

$$d_i = N_i l_{\boldsymbol{\tau}} + s_i(\boldsymbol{p}_i^N)$$

where N_i is the number of completed track loops and s_i is computed as in (4). Neglecting the constant terms, the objective function of player i is then to maximize the difference

$$f_i = s_i(\boldsymbol{p}_i^N) - s_j(\boldsymbol{p}_j^N). \tag{6}$$

Defining $\boldsymbol{\theta}_i = (\boldsymbol{p}_i^1, \dots, \boldsymbol{p}_i^N, \boldsymbol{u}_i^1, \dots, \boldsymbol{u}_i^N)$ and neglecting the second term in (6), which does not depend on player i's actions, the optimization problem for each robot can be rewritten as

$$\max_{\boldsymbol{\theta}_i} \; s_i(\boldsymbol{p}_i^N) \tag{7a}$$
$$\text{s.t.} \quad \boldsymbol{p}_i^k = \boldsymbol{p}_i^{k-1} + \boldsymbol{u}_i^k \tag{7b}$$
$$\left\|\boldsymbol{p}_j^k - \boldsymbol{p}_i^k\right\| \geq \overline{d}_i \tag{7c}$$
$$\left|\boldsymbol{n}(\boldsymbol{p}_i^k)^T [\boldsymbol{p}_i^k - \boldsymbol{\tau}(\boldsymbol{p}_i^k)]\right| \leq w_{\boldsymbol{\tau}} \tag{7d}$$
$$\left\|\boldsymbol{u}_i^k\right\| \leq \overline{u}_i. \tag{7e}$$

Because of the collision avoidance constraints (5), in order to calculate its optimal trajectory, each robot needs access to its opponent's strategy. However, since the robots are competing against each other, we do not expect them to share/communicate their plans. Instead, each robot needs to model the opponent and predict its actions. We believe that game theory [17] is the correct framework to describe this non-cooperative scenario. In particular, drone racing can be seen as a *zero-sum* differential game because clearly from (6) one has $f_1 + f_2 = 0$.

III. GAME THEORETIC FORMULATION

For simplicity of notation let us rewrite problem (7) in a more compact and general form

$$\max_{\boldsymbol{\theta}_i} \; s_i(\boldsymbol{\theta}_i) \tag{8a}$$
$$\text{s.t.} \; \boldsymbol{h}_i(\boldsymbol{\theta}_i) = 0 \tag{8b}$$
$$\boldsymbol{g}_i(\boldsymbol{\theta}_i) \leq 0 \tag{8c}$$
$$\boldsymbol{\gamma}_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \leq 0 \tag{8d}$$

where:

- $\boldsymbol{h}_i$ represents the equality constraints (7b) involving a single player;
- $\boldsymbol{g}_i$ represents the inequality constraints (7d) and (7e) involving a single player;
- $\boldsymbol{\gamma}_i$ represents the inequality constraints (7c) involving both players.

Let us also define $\Theta_i \subseteq \mathbb{R}^{4N}$ as the space of admissible strategies for player i, i.e. strategies that satisfy (8b) to (8d). Note that, due to (7c) and (8d), one has $\Theta_i = \Theta_i(\boldsymbol{\theta}_j)$, i.e. the strategy of one player determines the set of admissible strategies of its opponent and, as a consequence, can influence this latter's behavior.

A Nash equilibrium (see [17]) is a strategy profile $(\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^*) \in \Theta_1 \times \Theta_2$ such that no player can improve its own outcome by unilaterally changing its own strategy, i.e.

$$\boldsymbol{\theta}_i^* = \arg \max_{\boldsymbol{\theta}_i \in \Theta_i(\boldsymbol{\theta}_j^*)} s_i(\boldsymbol{\theta}_i), \forall i. \tag{9}$$

An alternative definition of Nash equilibria can be given by defining a *best reply map*

$$\mathcal{R}_i(\boldsymbol{\theta}_j) = \{\boldsymbol{\theta}_i \in \Theta_i(\boldsymbol{\theta}_j) : s_i(\boldsymbol{\theta}_i) = s_i^*(\boldsymbol{\theta}_j)\}$$

where

$$s_i^*(\boldsymbol{\theta}_j) = \max_{\boldsymbol{\theta}_i \in \Theta_i(\boldsymbol{\theta}_j)} s_i(\boldsymbol{\theta}_i) \tag{10}$$

is player i's *best-response return* to player j's strategy $\boldsymbol{\theta}_j$. One can show that a Nash equilibrium is a fixed point of the best reply map, i.e. such that $\boldsymbol{\theta}_i^* \in \mathcal{R}_i(\boldsymbol{\theta}_j^*)$.

Unfortunately, since problem (7) is not convex due to (7c), in general multiple Nash equilibria may exist (e.g. left vs right side overtaking). Additionally, computing the *exact* value of some Nash equilibrium generally requires numerical algorithms whose computational complexity makes them still not well suited for online robot control. Therefore, the next section describes an iterative algorithm that allows to *approximate* Nash equilibria in real time.

A. Numerical resolution of Nash equilibria

In order to approximate Nash equilibria in real time, we use an iterated best response algorithm (IBR). Starting from an initial guess of the Nash equilibrium strategy profile, we update each player's strategy, alternatively, to the best-response to the current opponent's strategy. This is done by solving a standard optimization problem in which one player strategy is allowed to change while the opponent's one is kept constant. Intuitively, if the resulting sequence of strategy profiles converges, it follows that each player is best-responding to its opponent. If this is the case, then no profitable unilateral change of strategy exists as required by the Nash equilibrium definition (9).

From our perspective, in the aim of developing a real time planner, IBR has the advantage that one can finely tune how much each player takes into account the reactivity of its opponent. By limiting the number of iterations per planning step, one can cover a spectrum of behaviors ranging from a very efficient, but naïve, classical optimal control problem (with a fixed guess for the opponent strategy) to a more computationally expensive but fully game theoretic approach.

Unfortunately, a direct application of IBR to (7) does not allow to fully capture the implications of the collision avoidance constraints (7c). As already mentioned, in fact, since player i has no direct influence over the final position of player j (i.e. s_j), the second term in (6) can be neglected in (7). However, since player j is calculating its strategy by solving an optimization problem similar to (7), due to the presence of the joint constraints (7c), player i does have an effect on $s_j^*(\boldsymbol{\theta}_i^*)$ (see the counterpart of (10) for player j). In other words, while player i does not affect player j's final position *in general*, it does affect it at the Nash equilibrium. To capture these effects, we substitute (8a) with the following cost function

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

where $\alpha \geq 0$ is a free parameter.

A closed form expression for $s_j^*(\boldsymbol{\theta}_i)$ is hard to obtain. Inspired by [24], we can, however, exploit sensitivity analysis to calculate a linear approximation around the current guess for the Nash equilibrium strategy profile.

Let us assume that, at the l-th iteration, a guess $\boldsymbol{\theta}_i^{l-1}$ for player i's strategy is available to player j. Given this strategy for its opponent, player j can solve the optimal control problem (7) with $\boldsymbol{\theta}_i = \boldsymbol{\theta}_i^{l-1}$ (fixed). This step will result in a new best-responding strategy for player j, $\boldsymbol{\theta}_j^l$, with the associated payoff $s_j^*(\boldsymbol{\theta}_i^{l-1})$. Assuming player i is now given the opportunity to modify its own strategy, we are interested in characterizing the variations of $s_j^*(\boldsymbol{\theta}_i)$ for $\boldsymbol{\theta}_i$ in the vicinity of $\boldsymbol{\theta}_i^{l-1}$ using a first-order Taylor approximation

$$s_j^*(\boldsymbol{\theta}_i) \approx s_j^*(\boldsymbol{\theta}_i^{l-1}) + \left.\frac{\mathrm{d}s_j^*}{\mathrm{d}\boldsymbol{\theta}_i}\right|_{\boldsymbol{\theta}_i^{l-1}} (\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{l-1}). \tag{11}$$

Exploiting the Karush–Kuhn–Tucker (KKT) necessary optimality conditions associated to player j's optimal control problem (8) one can prove the following result.

Lemma 1. *If s_j^* is the optimal value of an optimization problem obtained from* (8) *by exchanging subscripts i and j, then*

$$\left.\frac{ds_j^*}{d\boldsymbol{\theta}_i}\right|_{\boldsymbol{\theta}_i^{l-1}} = -\boldsymbol{\mu}_j^l \left.\frac{\partial \gamma_j}{\partial \boldsymbol{\theta}_i}\right|_{(\boldsymbol{\theta}_i^{l-1}, \boldsymbol{\theta}_j^l)} \tag{12}$$

where $\boldsymbol{\theta}_j^l \in \mathcal{R}_j(\boldsymbol{\theta}_i^{l-1})$ is the best-response of player j to $\boldsymbol{\theta}_i^{l-1}$ and $\boldsymbol{\mu}_j^l$ is the row vector of Lagrange multipliers associated to the joint inequality constraints (8d).

Proof. A full discussion on sensitivity analysis can be found in [25]. A brief proof, specific to the case at hand, is reported in Appendix A. □

Neglecting any term that is constant with respect to $\boldsymbol{\theta}_i$, we then propose that the ego vehicle solves the following optimization problem alternatively for itself and its opponent:

$$\max_{\boldsymbol{\theta}_i \in \Theta_i^l} s_i(\boldsymbol{\theta}_i) + \alpha \boldsymbol{\mu}_j^l \left.\frac{\partial \gamma_j}{\partial \boldsymbol{\theta}_i}\right|_{(\boldsymbol{\theta}_i^{l-1}, \boldsymbol{\theta}_j^l)} \boldsymbol{\theta}_i \tag{13}$$

where Θ_i^l respresents the space of strategies $\boldsymbol{\theta}_i$ that satisfy (8b) to (8d) with $\boldsymbol{\theta}_j = \boldsymbol{\theta}_j^l$.

Theorem 1. *If $\gamma_1(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \gamma_2(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ and the iterations converge to a solution $(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l)$, then the strategy tuple $(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l)$ satisfies the necessary conditions for a Nash equilibrium.*

Proof. Applying Karush-Kuhn-Tucker conditions to (8) one obtains the following set of necessary conditions for a Nash equilibrium $(\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^*)$ and the associated Lagrange multipliers

$$\frac{\partial s_i}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i^*) - \boldsymbol{\mu}_i^* \frac{\partial \gamma_i}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i^*, \boldsymbol{\theta}_j^*) - \boldsymbol{\lambda}_i^* \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i^*) - \boldsymbol{\nu}_i^* \frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i^*) = 0 \tag{14a}$$

$$\boldsymbol{h}_i(\boldsymbol{\theta}_i^*) = 0 \tag{14b}$$

$$\boldsymbol{g}_i(\boldsymbol{\theta}_i^*) \leq 0 \tag{14c}$$

$$\boldsymbol{\nu}_i^* \boldsymbol{g}_i(\boldsymbol{\theta}_i^*) = 0, \boldsymbol{\nu}_i^* \geq 0 \tag{14d}$$

$$\gamma_i(\boldsymbol{\theta}_i^*, \boldsymbol{\theta}_j^*) \leq 0 \tag{14e}$$

$$\boldsymbol{\mu}_i^* \gamma_i(\boldsymbol{\theta}_i^*, \boldsymbol{\theta}_j^*) = 0, \boldsymbol{\mu}_i^* \geq 0 \tag{14f}$$

Now assume that the iterative algorithm described in Sect. III converges to a solution $(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l)$, i.e. $\boldsymbol{\theta}_i^{l+1} = \boldsymbol{\theta}_i^l$ for both players. Then, by applying the KKT conditions to problem (13), $(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l)$ must satisfy

$$\frac{\partial s_i}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i^l) + \alpha \boldsymbol{\mu}_j^l \frac{\partial \gamma_j}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i^l, \boldsymbol{\theta}_j^l) - \boldsymbol{\mu}_i^l \frac{\partial \gamma_i}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i^l, \boldsymbol{\theta}_j^l) \boldsymbol{\lambda}_i^l \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i^l) - \boldsymbol{\nu}_i^l \frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i^l) = 0 \tag{15a}$$

$$\boldsymbol{h}_i(\boldsymbol{\theta}_i^l) = 0 \tag{15b}$$

$$\boldsymbol{g}_i(\boldsymbol{\theta}_i^l) \leq 0 \tag{15c}$$

$$\boldsymbol{\nu}_i^l \boldsymbol{g}_i(\boldsymbol{\theta}_i^l) = 0, \boldsymbol{\nu}_i^l \geq 0 \tag{15d}$$

$$\gamma_i(\boldsymbol{\theta}_i^l, \boldsymbol{\theta}_j^l) \leq 0 \tag{15e}$$

$$\boldsymbol{\mu}_i^l \gamma_i(\boldsymbol{\theta}_i^l, \boldsymbol{\theta}_j^l) = 0, \boldsymbol{\mu}_i^l \geq 0 \tag{15f}$$

If one additionally has $\frac{\partial \gamma_i}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i^l, \boldsymbol{\theta}_j^l) = \frac{\partial \gamma_j}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i^l, \boldsymbol{\theta}_j^l)$ (as it is the case for our problem), then one can see that $(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l)$ satisfy (14a) to (14e) with $\boldsymbol{\lambda}_i^* = \boldsymbol{\lambda}_i^l, \boldsymbol{\nu}_i^* = \boldsymbol{\nu}_i^l$ and $\boldsymbol{\mu}_i^* = \boldsymbol{\mu}_i^l - \alpha \boldsymbol{\mu}_j^l$. In order to satisfy (14f), however, one also needs to impose that:

$$(\boldsymbol{\mu}_1^l - \alpha \boldsymbol{\mu}_2^l)\gamma_1(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l) = 0 \tag{16a}$$

$$\boldsymbol{\mu}_1^l \geq \alpha \boldsymbol{\mu}_2^l \tag{16b}$$

$$(\boldsymbol{\mu}_2^l - \alpha \boldsymbol{\mu}_1^l)\gamma_2(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l) = 0 \tag{16c}$$

$$\boldsymbol{\mu}_2^l \geq \alpha \boldsymbol{\mu}_1^l. \tag{16d}$$

Using (15f), (16a) and (16c) reduce to

$$\alpha \boldsymbol{\mu}_2^l \gamma_1(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l) = 0$$
$$\alpha \boldsymbol{\mu}_1^l \gamma_2(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l) = 0.$$

Exploiting, again, (15f), this condition is satisfied if $\gamma_1(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l) = \gamma_2(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l)$ for all active constraints and if the sets of active constraints are the same for both players (i.e. $\boldsymbol{\mu}_1^l > 0 \iff \boldsymbol{\mu}_2^l > 0$). Both these conditions are satisfied if, as it is the case for our application, $\gamma_1(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l) = \gamma_2(\boldsymbol{\theta}_1^l, \boldsymbol{\theta}_2^l)$. As for (16b) and (16d), instead, if $\gamma_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \gamma_j(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$, one can enforce them by making α sufficiently small. Alternatively, one could also monotonically decrease α as the iterative best-response scheme progresses. □

In the drone racing scenario, in particular, using (4) and (5) after some straightforward calculation, (13) reduces to

$$\max_{\boldsymbol{\theta}_i \in \Theta_i^l} \left[\arg\min_s \tfrac{1}{2} \left\| \boldsymbol{\tau}(s) - \boldsymbol{p}_i^N \right\|^2 + \alpha \sum_{k=1}^N \mu_j^{k,l} \boldsymbol{\beta}_{ij}^{k,l^T} \boldsymbol{p}_i^k \right] \tag{18}$$

where

$$\boldsymbol{\beta}_{ij}^{k,l} = \frac{\boldsymbol{p}_j^{k,l} - \boldsymbol{p}_i^{k,l-1}}{\left\| \boldsymbol{p}_j^{k,l} - \boldsymbol{p}_i^{k,l-1} \right\|}.$$

To obtain a more intuitive interpretation of this result, let us assume that the track is linear and aligned to a unit vector $\boldsymbol{t}$ so that the first term in (18) can be rewritten as $\boldsymbol{t}^T \boldsymbol{p}_i^N$ (see Sect. III-B for details). Since player i cannot modify the strategy of player j, the following problem has the same solutions as (18)

$$\max_{\boldsymbol{\theta}_i \in \Theta_i^l} \boldsymbol{t}^T \boldsymbol{p}_i^N - \alpha \sum_{k=1}^N \mu_j^{k,l} \boldsymbol{\beta}_{ij}^{k,l^T} (\boldsymbol{p}_j^{k,l} - \boldsymbol{p}_i^k) \tag{19}$$

We can then notice the following insightful facts. First of all, if none of the collision avoidance constraints (7c) were active in the l-th instance of problem (7), i.e. if $\mu_j^{k,l} = 0$, then (19) reduces to (7). This has an intuitive explanation: if the collision avoidance constraints are not active, the optimal control problems for the two players are independent of each other and the original dynamic game reduces to a pair of classical optimal control problems. Clearly, in this case, the only sensible strategy for a player is to advance as much as possible along the track.

The problem becomes much more interesting when the collision constraints are active ($\mu_j^{k,l} > 0$). In this case, indeed, the cost function optimized in (19) contains additional terms with respect to (7c). By inspecting these terms, one can easily notice that they have a positive effect on player i's reward if robot i reduces its distance from player j's predicted position ($\boldsymbol{p}_j^{k,l}$) along the direction of $\boldsymbol{\beta}_{ij}^{k,l}$. The intuition behind this is that, when the collision avoidance constraints are active, player i can win the race by either going faster along the track or by getting in the way of player j, thus obstructing its motion.

Isolating the last term in the summation, one can also rewrite the problem as

$$\max_{\boldsymbol{\theta}_i \in \Theta_i^l} \left(\boldsymbol{t} + \alpha \mu_j^{k,N} \boldsymbol{\beta}_{ij}^{k,N}\right)^T \boldsymbol{p}_i^N + \alpha \sum_{k=1}^{N-1} \mu_j^{k,l} \boldsymbol{\beta}_{ij}^{k,l^T} \boldsymbol{p}_i^k.$$

From this alternative expression it is clear that, depending on the value of $\alpha\mu_j^{k,N}$, player i might actually find it more convenient to move its last position in the direction of player j ($\boldsymbol{\beta}_{ij}^{k,N}$) rather than along the track ($\boldsymbol{t}$). One can then also interpret the free scalar gain α as an *aggressiveness* factor. Note that player i can exploit this effect only so long as it does not cause a violation of its own collision avoidance constraint (7c).

Using (7b) one can also substitute $\boldsymbol{p}_i^N = \boldsymbol{p}_i^n + \sum_{k=n+1}^N \boldsymbol{u}_i^k$ and draw similar conclusions for any intermediate position $\boldsymbol{p}_i^n$.

Before concluding this section, we want to stress the fact that, since the players do not communicate with each other, each of them must independently run the iterative algorithm described above and alternatively solve the optimization problem (18) for themselves and for their opponent. In order to generate control inputs in real time, in our implementation we do not wait until convergence to a Nash equilibrium. Instead, we perform a constant number of iterations L, which can be set depending on the available computational resources. Since updating the opponent's strategy is only useful if this is exploited for recomputing a player's own strategy, we conclude each player's iterations with an extra resolution of its own optimal control problem. This also ensures that the resulting strategy profile satisfies the player's own constraints.

As for the resolution of each player's optimal control problem (18), we use an iterative algorithm described in Sect. III-B.

B. Numerical resolution of players' optimization

The resolution strategy described in Sect. III-A relies on the assumption that, at each iteration l, an optimal solution to problem (18) can be found given the current guess for the opponent strategy at the equilibrium.

Note that (18) is a well posed problem. Indeed, due to the system dynamic constraints (7b), the input boundedness imposed by constraints (7e), and assuming that the sampling time is finite, the set Θ_i^l is bounded. On the other hand Θ_i^l is also never empty because the solution $\boldsymbol{\theta}_i = (\boldsymbol{p}_i^0, \ldots, \boldsymbol{p}_i^0, \boldsymbol{0}_2, \ldots, \boldsymbol{0}_2)$ is always feasible, assuming that the robots do not start from a position that violates (7c) and (7d). Persistent feasibility is, therefore, guaranteed.

Unfortunately, problem (18) is also non-linear and non-convex. A source of non-convexity, in particular, is the reciprocal collision avoidance constraint (7c). For example, player i can potentially overtake player j by passing on the left or right side and, in some situations, these two solutions might even result in an equivalent payoff. Because of the aforementioned non-convexity, local optimization strategies might, in general, result in suboptimal solutions. In this work, however, we must calculate solutions to (18) in a very limited amount of time for online control. We then opt for a local optimization strategy thus potentially sacrificing optimality for increasing computational speed.

Assume that player i is at the l-th iteration of the Nash equilibrium search. The predicted strategy for player j is then $\boldsymbol{\theta}_j^l$ and it remains fixed while player i is solving problem (18), again, iteratively. In order to simplify the notation, in this section we drop the superscript l that indicates the Nash equilibrium search iteration and, instead, we use the superscript to indicate the *internal* iterations used to solve (18). Moreover, to clarify the notation even further, we use a $\bar{\cdot}$ accent to indicate all quantities that remain constant across all inner iterations used to solve a single instance of (18). Therefore, assume that player i's current guess of its optimal strategy is $\boldsymbol{\theta}_i^m$. We use $\boldsymbol{\theta}_i^m$ to compute a convex Quadratically-Constrained Linear (QCLP) approximation of problem (18).

Constraints (7b) and (7e) can be used as they are because they are either linear or quadratic and convex. The linear approximation of (7c) and (7d) is also straightforward and it results in the following constraints

$$\begin{aligned} \boldsymbol{\beta}_{ij}^{k,m^T}(\bar{\boldsymbol{p}}_j^k - \boldsymbol{p}_i^k) &\geq \bar{d}_i \\ \left|\boldsymbol{n}_i^{k,m^T}(\boldsymbol{p}_i^k - \boldsymbol{\tau}_i^{k,m})\right| &\leq w_{\boldsymbol{\tau}}, \end{aligned}$$

with $\boldsymbol{\beta}_{ij}^{k,m} = \frac{\bar{\boldsymbol{p}}_j^k - \boldsymbol{p}_i^{k,m}}{\|\bar{\boldsymbol{p}}_j^k - \boldsymbol{p}_i^{k,m}\|}$, $\boldsymbol{n}_i^{k,m} = \boldsymbol{n}(\boldsymbol{p}_i^{k,m})$, and $\boldsymbol{\tau}_i^{k,m} = \boldsymbol{\tau}(\boldsymbol{p}_i^{k,m})$.

The only term that requires some attention is the linear approximation of the cost function in (13) and, in particular, of its first term because we do not have a closed form expression for s_i as a function of $\boldsymbol{p}_i^N$. However, since $\boldsymbol{p}_i^N$ is a constant parameter in the optimization problem that defines s_i, we can exploit sensitivity analysis again to compute the derivative of s_i with respect to $\boldsymbol{p}_i^N$. To this end, let us rewrite

$$s_i = \arg\min_s d(s, \boldsymbol{p}_i^N), \text{with } d(s, \boldsymbol{p}_i^N) = \tfrac{1}{2}\left\|\boldsymbol{\tau}(s) - \boldsymbol{p}_i^N\right\|^2.$$

Then, as shown in [25] (and summarized in Appendix B for the case at hand) the derivative of s_i with respect to $\boldsymbol{p}_i^N$ can be calculated as

$$\frac{\mathrm{d}s_i}{\mathrm{d}\boldsymbol{p}_i^N} = -\left(\frac{\partial^2 d}{\partial s^2}\right)^{-1} \frac{\partial^2 d}{\partial s \partial \boldsymbol{p}_i^N} = \frac{\boldsymbol{\tau}'}{\|\boldsymbol{\tau}'\|^2 - \left(\boldsymbol{p}_i^N - \boldsymbol{\tau}\right)^T \boldsymbol{\tau}''}. \quad (20)$$

Exploiting the arc length parameterization and the relations (2) and (3) we conclude

$$\frac{\mathrm{d}s_i}{\mathrm{d}\boldsymbol{p}_i^N} = \frac{\boldsymbol{t}^T}{1 - \kappa\big(\boldsymbol{p}_i^N - \boldsymbol{\tau}\big)^T \boldsymbol{n}} := \boldsymbol{\sigma}(\boldsymbol{p}_i^N)$$

where $\boldsymbol{t}, \boldsymbol{n}$ and $\boldsymbol{\tau}$ must be computed for $s = s_i(\boldsymbol{p}_i^N)$. Neglecting any term that does not depend on $\boldsymbol{\theta}_i$, the cost function can then be approximated around $\boldsymbol{\theta}_i^m$ as

$$\boldsymbol{\sigma}(\boldsymbol{p}_i^{N,m})\boldsymbol{p}_i^N + \alpha \sum_{k=1}^{N} \overline{\mu}_j^k \overline{\boldsymbol{\beta}}_{ij}^{k\,T} \boldsymbol{p}_i^k .$$

The solution $\boldsymbol{\theta}_i^{m+1}$ to the approximate QCLP problem can then be used to build a new approximation of problem (18). The sequential QCLP optimization terminates when either a maximum number of iterations has been reached or the difference between two consecutive solutions, $r = \left\|\boldsymbol{\theta}_i^{m+1} - \boldsymbol{\theta}_i^m\right\|$, is smaller than a given threshold.

C. Alternative control strategies

In order to asses the effectiveness of our approach, in the simulations of Sect. IV, we let our controller compete with the following alternative control strategies.

1) Model predictive control (MPC): This strategy is based on the common, but naïve, assumption that player i's opponent will follow a straight line trajectory at (constant) maximum linear velocity along the local direction of the track, i.e. $\overline{v}_j \boldsymbol{t}(s(\boldsymbol{p}_j^0))$. Based on this assumption, player i can predict player j's strategy and solve (7) as a single classical optimal control problem. The numerical optimization scheme described in Sect. III-B can be used also in this case to efficiently compute a locally optimal solution.

2) Reciprocal velocity obstacles (RVO): This second benchmark strategy is based on the multi-agent collision avoidance library proposed in [26] and implemented in the open-source library RVO2[1]. We approximated the boundaries of the track with a set of polygonal obstacles. For each robot, RVO uses a reference linear velocity with maximum norm $\overline{v}_i$ and direction computed as $\boldsymbol{t}(\boldsymbol{p}_i^0) + \rho(\boldsymbol{\tau}(s_i^0) - \boldsymbol{p}_i^0)$ where $\rho > 0$ is a free parameter that allows to trade off between the first term, which makes the robot follow the local direction of the track, and the second one, which keeps the robot close to the center line.

IV. RESULTS

A. Simulations

In order to validate our approach, we performed an extensive simulation campaign. We used the open-source RotorS package [27] to simulate the full quadrotor dynamics. Our planning algorithm was implemented in C++ and interfaced with the simulator using ROS. We used a simulation time step of $10\,\mathrm{ms}$, but we run our planners at $20\,\mathrm{Hz}$. We also used state-of-the-art nonlinear controllers [28] to drive our quadrotors along the optimal trajectory resulting from the solution of (18).

The two simulated robots have a radius of $0.3\,\mathrm{m}$ and maintain a minimum relative distance $\overline{d}$ of $0.8\,\mathrm{m}$ from their opponent. The simulated track is represented in Fig. 2. The track fits into a $15\,\mathrm{m} \times 11\,\mathrm{m}$ rectangle and its half-width $w_{\boldsymbol{\tau}}$ is $1.5\,\mathrm{m}$. The origin of the world frame was set at the center of the longest straight segment of the track.

We let our game-theoretic planner and the alternative strategies described in Sect. III-C compete against each other over multiple races differing by the robots initial positions. In order to enforce some interaction, we set the maximum linear velocities of the two robots to $0.5\,\mathrm{m/s}$ and $0.6\,\mathrm{m/s}$ and we made the faster robot always start behind the slower one. In particular, for each race, we sampled the initial position of the faster robot from a uniform distribution in the rectangle $[-0.1, 1.5] \times [-0.7, 0.7]$. Similarly, the position of the slower robot was sampled from the rectangle $[1.6, 1.7] \times [-0.7, 0.7]$. We discarded any pair of sampled initial positions that would violate the collision avoidance constraints (5).

We ran a total of 900 simulations in which the same 150 sampled initial conditions were used for each of the following scenarios:

case I: fast GTP vs. slow MPC;
case II: fast MPC vs. slow GTP;
case III: fast GTP vs. slow RVO;
case IV: fast RVO vs. slow GTP;
case V: fast MPC vs. slow RVO;
case VI: fast RVO vs. slow MPC.

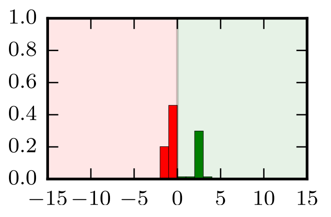
Here, and in the rest of this section, the acronym GTP indicates the Game Theoretic Planner developed in this paper. We terminated each simulation as soon as one robot completed an entire track loop and reached the finish line positioned at $x = 2.32\,\mathrm{m}$.

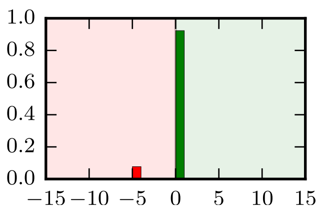
In Fig. 3, we report an histogram representation of the final distance along the track (i.e. the arch-length difference (6)) between the two robots. In cases I to IV, the distance is calculated in such a way that it is positive when the robot controlled using GTP wins the race and negative otherwise. In cases V and VI, instead, a positive distance indicates a victory for the MPC planner over RVO. A green and red coloring is also used to highlight positive and negative parts of the histogram.

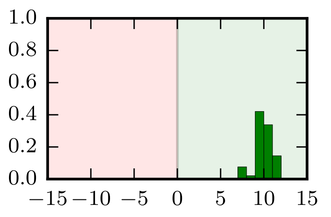
In Fig. 4, we report the position traces for the two competing robots for all of the simulations. The traces are divided by case and we used the following color code: GTP – green, MPC – blue, RVO – red.

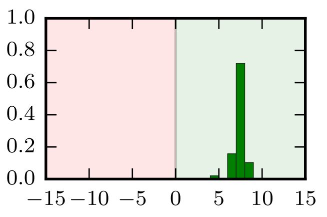
First of all, as it can be seen from Figs. 3c to 3f, the RVO strategy is clearly the least effective one among the three alternatives considered in this paper. Regardless of the initial placing and of the possible advantage in terms of maximum speed, the robots controlled with this strategy lost all races. We believe that the main reason for this poor performance is the fact that RVO is a reactive (instantaneous) control strategy while both GTP and MPC use an extended planning horizon. Because of the lack of planning, RVO does not *anticipate* (and appropriately cut) the curves and ends up following a longer trajectory. This can clearly be noticed by looking at Figs. 4c to 4f. The performance could possibly be improved by considering alternative heuristics in the calculation of the reference/desired velocity for the RVO algorithm.

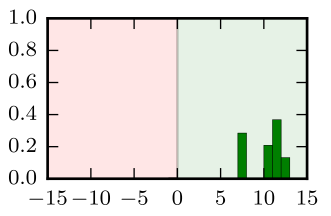
[1]http://gamma.cs.unc.edu/RVO2/


(a) Case I: fast GTP vs. slow MPC


(b) Case II: fast MPC vs. slow GTP


(c) Case III: fast GTP vs. slow RVO


(d) Case IV: fast RVO vs. slow GTP


(e) Case V: fast MPC vs. slow RVO

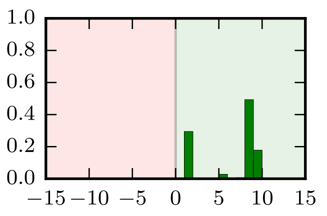
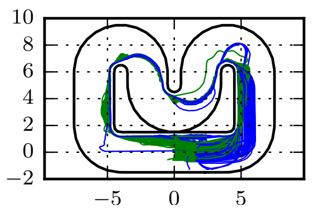
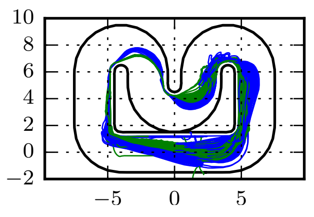
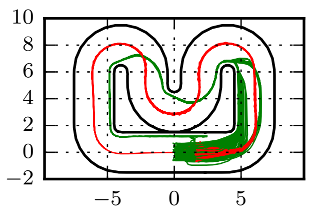
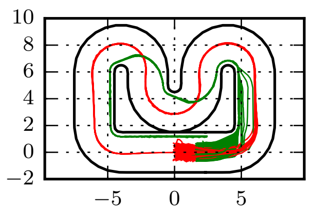

(f) Case VI: fast RVO vs. slow MPC

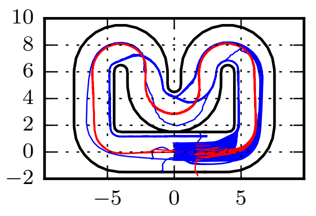
Figure 3: Histogram representation of the final arch-length difference (as in (6)) between the two robots for all of the simulations. The simulations are divided by case as indicated in the captions. In (a) to (d), the distance is calculated in such a way that it is positive when the robot controlled using GTP wins the race and negative otherwise. In (e) and (f), instead, a positive distance indicates a victory for the MPC planner over RVO. A green and red coloring is also used to highlight positive and negative parts of the histogram.


(a) Case I: fast GTP vs. slow MPC


(b) Case II: fast MPC vs. slow GTP


(c) Case III: fast GTP vs. slow RVO


(d) Case IV: fast RVO vs. slow GTP


(e) Case V: fast MPC vs. slow RVO

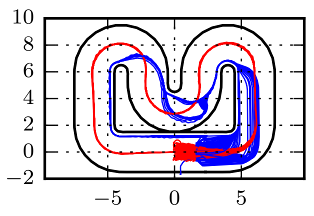

(f) Case VI: fast RVO vs. slow MPC

Figure 4: Position traces for the two competing robots for all of the simulations. The simulations are divided by case as indicated in the captions and the following color code was used for the planners: GTP – green, MPC – blue, RVO – red.

The comparison between GTP and MPC is somewhat more fair because both strategies effectively follow the track and the only difference between the two lies in the way they interact with the opponent (the two algorithms are perfectly identical when the two robots do not interact). A direct comparison between the two strategies is provided by Figs. 3a and 3b. From Fig. 3a, we can notice that, when the drone running the GTP planner is faster than the MPC one, it manages to overtake the MPC planner, which starts from an advantageous position, in approximately 30% of the races. A closer look at the simulations reveals that, in this scenario, the GTP planner often tends to "overestimate" its opponent (it assumes the opponent is using GTP as well). Consequently, when attempting to overtake, it expects the opponent to block its motion and ends up following an overly cautions trajectory moving sideways along the track more than necessary.

On the other hand, when the GTP is slower, it manages to defend its initial advantage for the vast majority of the races (see Fig. 3b). Looking at Fig. 4b, we can clearly visualize the strategy adopted by the GTP planner to defend its position, especially towards the end of the race (the bottom straight part of the track). The GTP planner clearly moves sideways along the track to block the MPC planner thus exploiting the collision avoidance constraint to its own advantage. The MPC planner, instead, cannot adopt a similar strategy because it does not properly model the reactions of its opponent. On the contrary, by assuming that the opponent will move straight along the path, completely careless of possible collisions, the MPC planner is often forced to make room to the opponent because of its own collision avoidance constraints (see, for example, the blue traces in the top right part of Fig. 4a).

An indirect comparison between GTP and MPC can also be done by analyzing how they both perform against RVO in similar situations. Both when competing against a slower robot (see Figs. 3c and 3e) and against a faster one (see Figs. 3d and 3f) the GTP planner tends to win with a slightly larger separation in average and a much more narrow distribution of final distances. The improvement is more significant when the GTP is playing in a "defensive" role, i.e. it is controlling a slower robot with an initial advantage. We believe that this is due, once again, to an overly cautious behavior when attempting to overtake the opponent in cases I and III.

To fully appreciate the interactions between the robots, we encourage the reader to watch the video available at [16] showing a simulated race. In this simulation, both robots were running the game-theoretic planner described in this paper. Finally, we have conducted preliminary hardware experiments with two quadrotor UAVs in the Robotics and Perception Group at the University of Zurich (see Fig. 1). The UAVs use onboard monocular vision to estimate their opponent's relative position, and compute their racing trajectories online using our algorithm. A video is available at [16], and a more detailed paper with comprehensive hardware experiments is forthcoming.

V. CONCLUSIONS

In this paper we described a novel online motion planning algorithm for two-player drone racing. By exploiting sensitivity analysis within an iterated best response algorithm, our planner can effectively model and even exploit the opponent's reactions.

From a theoretical point of view, we showed that, if the iterative resolution strategy converges to a solution, then this latter satisfies necessary conditions for a Nash equilibrium. Moreover, we demonstrated the effectiveness of our approach through an extensive set of simulations in which our planner was let to compete against two alternative, and well established, approaches. Finally, our planning strategy can run in real time and we are currently conducting some experimental tests on real hardware. Some videos are available at [16].

Despite the encouraging results, our planner still presents some weakness. First of all, because of the non-convexity of our problem, the optimization algorithm can converge to local minima. In the future, we want to investigate the use of mixed integer approaches to better handle the non-convex constraints of our problem. In addition to this, our planner assumes that the opponent is using a similar planning strategy with a known cost function. Even if our algorithm already shows very good performance when competing against alternative strategies (which clearly violate this assumption), it would be interesting to couple our strategy with an online approach for learning the opponent policy and/or cost function. Finally, we also plan to extend our results to races involving more than two players. A first possibility, in this case, would be to simply introduce an additional sensitivity term and an optimal control problem for each of the added opponents and iterate over all players until convergence. Another alternative would be for each player to consider all of the opponents as a single agent. This would reduce computation but possibly induce more conservative behaviors due to the underlying assumption of cooperation among the opponents.

APPENDIX

A. *Proof of Lemma 1*

In order to simplify the notation as much as possible, in this subsection we consider a streamlined form for the optimization problem of the form

$$\max_x \; s(x) \text{ s.t. } \gamma(x,c)=0 \tag{21}$$

where c is a scalar parameter and s and γ are scalar differentiable functions of their arguments. For each value of c, let us indicate with $x^*(c)$ the solution of (21) and with $s^*(c)=s(x^*(c))$ the associated optimal outcome. We want to study how the optimal cost s^* changes when c changes around a point $\overline{c}$, i.e.

$$\left.\frac{\mathrm{d}s^*(c)}{\mathrm{d}c}\right|_{\overline{c}} = \left.\frac{\mathrm{d}s(x^*(c))}{\mathrm{d}c}\right|_{\overline{c}} = \left.\frac{\mathrm{d}s(x)}{\mathrm{d}x}\right|_{x^*(\overline{c})} \left.\frac{\mathrm{d}x^*(c)}{\mathrm{d}c}\right|_{\overline{c}} \tag{22}$$

Since, for all c, $x^*(c)$ is an optimal solution to (21), it must satisfy the KKT necessary optimality conditions associated to (21), i.e.

$$\left.\frac{\mathrm{d}s(x)}{\mathrm{d}x}\right|_{x^*} - \mu \left.\frac{\partial\gamma(x,c)}{\partial x}\right|_{x^*} = 0 \tag{23}$$

$$\gamma(x^*(c),c) = \gamma^*(c) = 0 \tag{24}$$

where μ is the Lagrange multiplier associated to the equality constraint. Isolating the first term in (23) and substituting it in (22) we obtain

$$\left.\frac{\mathrm{d}s^*(c)}{\mathrm{d}c}\right|_{\overline{c}} = \mu \left.\frac{\partial\gamma(x,c)}{\partial x}\right|_{x^*(\overline{c})} \left.\frac{\mathrm{d}x^*(c)}{\mathrm{d}c}\right|_{\overline{c}}. \tag{25}$$

Note that, since (24) must remain true for all c, its total derivative w.r.t. c must also be zero, i.e.

$$\frac{\mathrm{d}\gamma^*(c)}{\mathrm{d}c} = \left.\frac{\partial\gamma(x,c)}{\partial x}\right|_{x^*} \frac{\mathrm{d}x^*(c)}{\mathrm{d}c} + \left.\frac{\partial\gamma(x,c)}{\partial c}\right|_{x^*} = 0. \tag{26}$$

Isolating the first term from (26) and substituting it in (25), we finally conclude that

$$\left.\frac{\mathrm{d}s^*(c)}{\mathrm{d}c}\right|_{\overline{c}} = -\mu \left.\frac{\partial\gamma(x,c)}{\partial c}\right|_{x^*(\overline{c})},$$

which reduces to (12) for $x=\boldsymbol{\theta}_j$, $\overline{c}=\boldsymbol{\theta}_i^{l-1}$, and $x^*=\boldsymbol{\theta}_j^l$.

This proof can trivially be extended to problems with multiple joint constraints or with additional constraints that do not depend on c (their derivatives with respect to c will simply be null). If the problem contains inequality constraints, instead, under the assumption that, in the vicinity of c, the set of active constraints remains the same, the proof can readily be applied by just considering an equivalent problem in which any active inequality constraint is transformed into an equality constraints and any inactive constraint is ignored.

B. *Proof of* (20)

Consider the following optimization problem:

$$\min_s d(s, \boldsymbol{p}_i^N).$$

We can interpret $\boldsymbol{p}_i^N$ as a constant parameter and study how the solution s_i to the above problem changes when $\boldsymbol{p}_i^N$ changes around a point $\overline{\boldsymbol{p}}_i^N$. Under the optimality assumption, for each value $\boldsymbol{p}_i^N$, the corresponding solution $s_i^N = s_i(\boldsymbol{p}_i^N)$ must satisfy the following necessary condition

$$\left.\frac{\partial d(s,\boldsymbol{p}_i^N)}{\partial s}\right|_{s_i^N} = 0. \tag{27}$$

Note that the left hand side of (27) is a function of $\boldsymbol{p}_i^N$ only and it must be zero for all $\boldsymbol{p}_i^N$. Therefore, its derivative with respect to $\boldsymbol{p}_i^N$ must also be zero

$$0 = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{p}_i^N} \left.\frac{\partial d(s,\boldsymbol{p}_i^N)}{\partial s}\right|_{s_i^N} = \left.\frac{\partial^2 d(s,\boldsymbol{p}_i^N)}{\partial s^2}\right|_{s_i^N} \frac{\mathrm{d}s_i^N}{\mathrm{d}\boldsymbol{p}_i^N} + \frac{\partial^2 d(s,\boldsymbol{p}_i^N)}{\partial s \partial \boldsymbol{p}_i^N}.$$

We can, then, conclude that:

$$\frac{\mathrm{d}s_i^N}{\mathrm{d}\boldsymbol{p}_i^N} = -\left[\left.\frac{\partial^2 d(s,\boldsymbol{p}_i^N)}{\partial s^2}\right|_{s_i^N} \right]^{-1} \frac{\partial^2 d(s,\boldsymbol{p}_i^N)}{\partial s \partial \boldsymbol{p}_i^N}$$

q.e.d.

References

[1] "World drone prix," Dubai, UAE, Mar. 2016. [Online]. Available: https://youtu.be/gc3_wEB9wnI

[2] H. Moon, Y. Sun, J. Baltes, and S. J. Kim, "The IROS 2016 competitions [competitions]," *IEEE Robotics and Automation Magazine*, vol. 24, no. 1, pp. 20–29, 2017.

[3] S. Jung, S. Cho, D. Lee, H. Lee, and D. H. Shim, "A direct visual servoing-based framework for the 2016 IROS autonomous drone racing challenge," *Journ. of Field Robot.*, vol. 35, no. 1, pp. 146–166, 2018.

[4] N. R. Kapania, J. Subosits, and J. C. Gerdes, "A sequential two-step algorithm for fast generation of vehicle racing trajectories," *Journ. of Dynamic Systems, Measurement, and Control*, vol. 138, no. 9, pp. 091 005–091 005–10, 2016.

[5] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE Int. Conf. on Robotics and Automation*, Stockholm, Sweden, May 2016, pp. 1433–1440.

[6] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic MPC for model-based reinforcement learning," in *2017 IEEE Int. Conf. on Robotics and Automation*, Singapore, May 2017.

[7] Yamaha Motor and SRI International, "Yamaha MOTOBOT 2," 2017. [Online]. Available: https://youtu.be/BjZPvXKewFk

[8] R. Stelzer and T. Pröll, "Autonomous sailboat navigation for short course racing," *Robotics and Autonomous Systems*, vol. 56, no. 7, pp. 604–614, 2008.

[9] A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *Int. Journ. on Robotics Research*, vol. 4, no. 3, pp. 109–117, 1985.

[10] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. Journ. on Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[11] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. Journ. on Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[12] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journ. of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[13] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. Journ. on Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[14] 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.

[15] F. Feurtey and T. Chikayama, "Simulating the collision avoidance behavior of pedestrians," Master's thesis, University of Tokyo, 2000.

[16] "Multi-robot Systems Lab (MSL) at Stanford University." [Online]. Available: https://msl.stanford.edu/game-theoretic-planning-autonomous-drone-racing

[17] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1998.

[18] T. Başar and P. Bernhard, *H-Infinity Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach*. Boston, MA, USA: Birkhäuser Boston, 2008.

[19] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, 2005.

[20] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Robotics: Science and Systems*, Cambridge, MA, USA, Jul. 2016, p. n/a.

[21] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Daejeon, Korea, Oct. 2016, pp. 66–73.

[22] A. Dreves and M. Gerdts, "A generalized Nash equilibrium approach for optimal control problems of autonomous cars," Optimal Control Applications and Methods, Jul. 2017, to be published.

[23] A. Liniger and J. Lygeros, "A non-cooperative game approach to autonomous racing," *arXiv preprint arXiv:1712.03913*, 2017.

[24] T. Raivio and H. Ehtamo, "On the numerical solution of a class of pursuit-evasion games," in *Advances in Dynamic Games and Applications*, ser. Annals of the International Society of Dynamic Games. Birkhäuser Boston, 2000, pp. 177–192.

[25] A. V. Fiacco, *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Academic Press, 1983.

[26] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *14th Int. Symp. on Robotics Research*, Lucerne, Switzerland, Aug. 2011, pp. 3–19.

[27] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Robot operating system (ros)," *Studies Comp.Intelligence Volume Number:625*, vol. The Complete Reference (Volume 1), no. 978-3-319-26052-5, p. Chapter 23, 2016, iSBN:978-3-319-26052-5.

[28] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3D mapping with a quadrotor MAV," *Journ. of Field Robot.*, vol. 33, no. 4, pp. 431–450, 2016.