BuzzRacer: A Palm-sized Autonomous Vehicle Platform for Testing Multi-Agent Adversarial Decision-Making

Zhiyuan Zhang[1] and Panagiotis Tsiotras[2]

***Abstract*— We present BuzzRacer, a palm-sized autonomous vehicle platform suitable for multi-agent autonomous racing. BuzzRacer consists of two parts. First, a software framework with multiple racetrack environments, dynamic simulation, visualization, and control pipelines. Second, a miniature autonomous vehicle platform capable of 1g acceleration and 3.5m/s top speed. BuzzRacer is an open-source project currently used at Georgia Tech in a project-based robotics course and research projects for experimental validation and benchmarking of novel planning and control algorithms.**

I. INTRODUCTION

A. Background

Autonomous racing presents a unique challenge of controlling an underactuated nonlinear system near control saturation while operating in close proximity to the track boundaries. In addition, algorithms must also interact with hard-to-predict opponents and satisfy stringent real-time constraints on solution time. The problem involves many challenges in the area of autonomy and has therefore been a topic of intense research in recent years [1]. In single-car racing, domain-specific hierarchical controllers [2], various Model Predictive Controller schemes (MPC) [3], [4], [5], and reinforcement-learning (RL) based methods [6] have achieved near or superhuman performance. In multi-car racing, game-theoretic approaches such as SE-IBR [7], ALGAMES [8], RL methods [9] have demonstrated promising results with interesting self-emerged dynamic behavior in blocking and overtaking similar to those observed by human racecar drivers.

Contrary to autonomous driving on public roads, autonomous racing takes place on a dedicated racetrack, a much more structured environment without heterogeneous traffic participants. In addition, autonomous racing operates a vehicle at its performance limit and has a much higher tolerance for risk. Despite these differences, autonomous racing can be a valuable validation environment to develop algorithms that may be adapted for autonomous driving. Experience in driving at the handling limit builds insights for controlling the vehicle at regions of the state space rarely visited during normal driving, making these regions available in emergency scenarios, where aggressive maneuvering is necessary to maintain safety (e.g., swerving to avoid a road hazard). A study on game theory's application to vehicle interaction can help future self-driving cars interact with humans more effectively, preventing the frozen robot problem [10] that stymies modern self-driving cars.

In spite of the recent surge in interest and the promising prospects, publications with experiments on physical platforms in the field of autonomous car racing remain sparse due to the high cost and effort in developing a suitable platform and solving the perception, localization, communication, and other problems prerequisite to on-board planning and control. The simulations and experiments used to validate existing game-theoretic car racing controllers are done in vastly dissimilar environments, and fair performance comparison is difficult. We believe the lack of a suitable platform contributes to this situation. In the following section, we categorize some existing popular autonomous vehicle platforms and discuss why they may not be ideal as a universal platform for testing autonomous racing algorithms.

B. Related Work

Most of the current literature makes use of scaled platforms. These are built from hobby remote-control (RC) cars. Georgia Tech's AutoRally uses a 1/5 scale RC car outfitted with a full desktop computer with graphics cards, stereo cameras, and differential GPS [11]. The MIT Racecar [12] and the Berkeley Autonomous Vehicle make use of a 1/10 scale chassis carrying a System-on-board (SOB) computer like an NVIDIA Jetson, and sensors including 2D LIDAR, camera, wheel-speed sensor, etc. These platforms are relatively affordable and suitable for full-stack projects involving perception, localization, planning, and control. However, they also have some limitations.

First, computation is done onboard, which means that researchers have to maintain a development environment for every vehicle they use and must work under the constraint of limited on-board computation resources. Many algorithms in game-theoretic control are computationally heavy and may not run on embedded hardware.

Second, the sensors, the computer, and the computer battery are heavy, negatively affecting the vehicle's dynamic performance. While useful for perception and localization, these components are redundant for control and planning algorithms that focus elsewhere.

Third, these vehicles need a sizable room to operate at a speed that generates interesting limit handling behavior like understeering and oversteering. Such spaces may be hard

This work is sponsored by ONR award N00014-18-2375 and NSF award 1849130.

[1] School of Aerospace Engineering, Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA, Email: zzhang615@gatech.edu

[2] School of Aerospace Engineering, Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA, Email: tsiotras@gatech.edu

to secure or are accessible only temporarily, and complex circuits may be difficult to reproduce accurately.

Observing a gap in the current autonomous vehicle platforms, we developed BuzzRacer, a 1/28 scale palm-sized vehicle platform capable of 1g lateral acceleration and 3.5m/s top speed. It supports two-way communication via WiFi and is equipped with a 6-axis inertial measurement unit (IMU). BuzzRacer is intended for offboard computing and utilizes a visual tracking system for state measurements. In addition to the hardware platform, we developed a modular software stack with racetrack creation, dynamic simulation, visualization, control pipelines, and extensions for additional functionalities. This software framework can be used for simulation, performance evaluation, algorithm comparison, and on-car experiments.

To the best of the authors' knowledge, only the ORCA platform [13] developed by ETH is analogous to the proposed BuzzRacer platform . However, ORCA is a closed-source platform and the dNano RC car that it is based on has been discontinued. BuzzRacer is open-source [14] and its base chassis is widely available.

Using BuzzRacer, we compared two state-of-the-art game-theoretic racing controllers, Iterated Best Response (IBR) and iLQGame, and their non-game-theoretic counterparts, Model Predictive Control (MPC) and iLQR.

The primary contribution of this paper is the proposed software and hardware platform; a secondary contribution is the comprehensive comparison of several game-theoretic controllers using this platform illustrating the utility of BuzzRacer as an inexpensive simulation and experimental platform.

The rest of the paper is organized as follows: Section II introduces the vehicle design. Section III discusses the software stack. Section IV details the implementation for the four controllers. Section V discusses simulation and experimental results, followed by an outlook of future work on the platform in Section VI.

II. Vehicle Design

Three major components work together under a local area network (LAN) during BuzzRacer operation. An Optitrack computer runs the proprietary visual tracking software and streams state measurements to the control computer(s). The control computer(s) execute the planning and control algorithms, and send commands to the vehicles. The vehicles execute actuation commands and report sensor measurements to the control computer(s). One control computer may control multiple vehicles or a separate control computer may be used for each vehicle if running a computationally expensive algorithm. Figure 1 shows the topology of these components.

A. Chassis

The underlying chassis of BuzzRacer is based on the Kyosho Mini-z RC car, a 1/28 scale rear wheel drive chassis popular with amateur RC enthusiasts (Figure 2). The realistic body shell attached to the chassis is retained for aesthetics and crash protection. Spherical visual tracking markers are affixed to the body shell. The BuzzRacer has dimensions of 165×80×40 mm and weighs 170 grams, just 4 grams heavier than the base chassis, allowing the vehicle to keep its original high-performance vehicle dynamics.

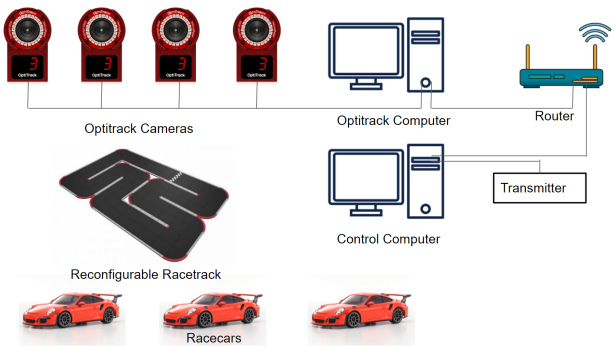

Fig. 1. BuzzRacer system components.

Pi Group	BuzzRacer	Full-sized
$\ell_f \ell^{-1}$	0.47	0.45
$t\ell^{-1}$	0.45	0.55
$C_\alpha \ell m^{-1} u^{-2}$	0.33	0.17
$I_{zz} m^{-1} \ell^{-2}$	0.24	0.25

TABLE I
Pi Group Dimension Analysis

Fig. 2. A pair of BuzzRacer platforms.

B. Dimension Analysis

To evaluate the similarity between our scaled system and a full-sized vehicle, we compare the nondimensional Pi Group in Table I [15], where ℓ_f denotes distance between CG and front axle, ℓ denotes wheelbase, t denotes track width, m denotes mass, C_α denotes tire stiffness, I_{zz} denotes rotational inertia in the vertical axis, and u denotes operating speed. The parameters are determined experimentally, with C_α measured with a banked slope experiment, and I_{zz} measured with a suspended trifilar torsion pendulum.

In our investigation of dimensional analysis, it is observed that all nondimensional Pi Group parameters exhibit proximity within one order of magnitude to each other. This alignment signifies that the proposed platform is kinematically and dynamically similar to its full-sized counterpart.

C. Embedded System

To enable communication over WiFi and control of the low-level actuator dynamics, the proprietary onboard electronics are replaced with a custom PCB. The replacement PCB consists of an Arduino Nano 33 IoT and two H-bridge ICs, one for drive motor control and another for steering servo motor control. Figure 3 shows a close-up of the PCB. For most RC platforms, steering is controlled via a standard servo. A PWM-encoded target angle is conveyed to the servo input and the servo control board uses a black-box closed-loop controller to track the commanded angle. A typical servo can take about 0.10 seconds to move 60 degrees, thus for high-performance applications, the steering servo dynamics is not trivial. In the BuzzRacer, the servo control circuitry is integrated into the main PCB, thus our firmware has access to measured steering angle and control over the low-level actuator dynamics. This improves the accuracy of our vehicle model.

The Arduino Nano 33 IoT is equipped with a SAMD21 Cortex-M0 32bit MCU at 48MHz, a u-blox NINA radio module providing WiFi and Bluetooth communication, and a 6-axis IMU. To control the drive motor and the steering motor, two 15kHz PWM signal pairs are generated using SAMD 21 IC's CMSIS function. The servo motor is controlled by a PID controller at 500Hz, and the duty cycle of the drive motor is mapped directly to the throttle command from upper-level software.

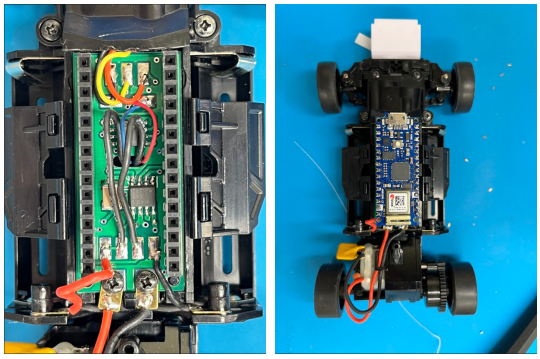

Fig. 3. Left: Replacement PCB. Right: BuzzRacer without body shell.

D. Communication

The SAMD MCU connects to a LAN via WiFi on startup. It then listens for incoming packets, and streams sensor measurements from the servo potentiometer, battery voltage, motor current, and IMU. This information is communicated via a lightweight protocol over UDP. The protocol contains fields for the sequence number, timestamp, destination address, source address, packet type, and payload. The payload field is padded for a fixed packet length of 64 bytes, padding length varies depending on package types, for instance, quality of service inquiries, actuator commands, sensor updates, and parameter settings. The communication frequency between the control computer and the onboard MCU is 100Hz, which is the maximum control frequency of the BuzzRacer platform.

III. Software Design

The BuzzRacer software stack consists of modules for various roles including simulation, planning, control, visualization, logging, etc. In order to run an experiment, the user creates a config file specifying the module to be loaded for each role and the parameters to be set for each module. This versatile design allows users to create different experiments easily without editing any source code. For example, a user may develop a controller in simulation, and simply change the experiment type from simulation to real world in the config file to run a physical experiment.

In this section, we introduce the core modules of our software stack. Figure 4 illustrates the relationship between the modules at runtime.

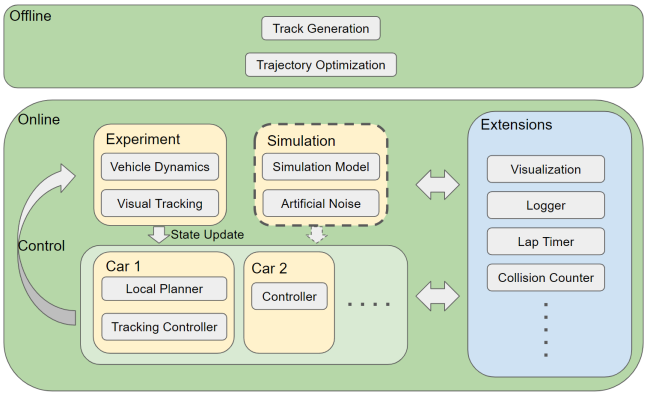

Fig. 4. BuzzRacer software stack.

A. Track

The main BuzzRacer track is assembled from straight and turn grids. We designed the track representation following the semantics of the physical race track layout. By specifying the sequence of each element, it is possible to create any track layout with a single line of source code. Figure 5 shows several tracks of varying complexity.

In addition to the grid-style track, users may also create arbitrary tracks by specifying track centerline and track width.

B. State Update

State updates come from two sources. In simulation, new state updates are calculated using a simulation model, which

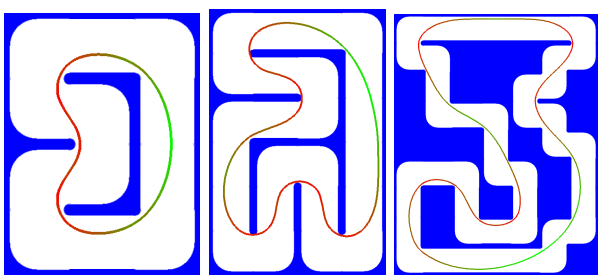

Fig. 5. Example grid-style tracks with reference trajectory.

is a single-track dynamic bicycle model with parameters from Unscented Kalman Filter (UKF) estimation [16]. In the actual experiments, the vehicle position and orientation are provided by a visual tracking system, and an Extended Kalman Filter (EKF) is used to estimate state derivatives. The BuzzRacer codebase supports both Optitrack and Vicon visual tracking systems. It is possible to add artificial noise to state updates to test the robustness of the controller.

C. Extensions

In addition to the basic control pipeline, extensions can be loaded at runtime to provide enhanced functionalities. For example, logging vehicle states and controls, visualizing cars on track in real-time, and recording lap times.

A configuration file written in xml syntax specifies which extensions are to be loaded at runtime.

IV. Controller Design

In this section, we describe the formulation of the four controllers benchmarked in our experiments.

A. Problem Formulation

Given a track defined by its reference path $\mathbf{r}(s) = (r_x, r_y)$ and boundaries $\rho_L(s), \rho_R(s)$, we use a discrete-time kinematic bicycle model described in Frenet coordinates [17] for the system dynamics.

Let $x = [s, v, n, \phi, \beta]^{\mathrm{T}}$ be the agent state, with s the progress along the reference curve, v the speed, n the lateral deviation,ϕ the heading deviation from the reference curve, and β the angle between the velocity vector and vehicle orientation. To simplify the constraints on vehicle acceleration, we select $u = [a_y, a_x]$, the lateral and longitudinal acceleration of an agent as control variables.

$$\dot{s} = \frac{v\cos(\phi)}{1 - n\kappa(s)}, \tag{1a}$$

$$\dot{v} = a_x\cos(\beta) + a_y\sin(\beta), \tag{1b}$$

$$\dot{n} = v\sin(\phi), \tag{1c}$$

$$\dot{\phi} = \dot{\beta} + \frac{v}{l_r}\sin(\beta) - \frac{v\cos(\phi)\kappa(s)}{1 - r\kappa(s)}, \tag{1d}$$

$$\dot{\beta} = (-a_x\sin(\beta) + a_y\cos(\beta))/v, \tag{1e}$$

where $\kappa(s)$ is the signed curvature of the reference curve at progress s. We discretized the model with Euler approximation.

For a non-game-theoretic agent, the control objective is to maximize progress over the horizon $k \in \{0, \dots, K\}$. For a game-theoretic agent, the game objective is to maximize the ego agent's lead over its opponents over the specified horizon. We use a coefficient α (Eq. (3a)) to tune the agent behavior, where a larger α corresponds to a more aggressive agent less sensitive to collision. When $\alpha > 0$, the agents play a competitive game, and when $\alpha < 0$, the agents play a coordination game.

The controllers need to keep the agents within the track boundaries and respect vehicle dynamics limits. The maximum longitudinal acceleration is bounded by the available motor torque as a function of the current vehicle speed, and the maximum overall acceleration is bounded by an elliptical traction circle. Finally, the agents should avoid collision with each other. These constraints are summarized in Eq. (2a)-(2d) below

$$\text{Track boundary } n < \rho_L(s), -n < \rho_R(s), \tag{2a}$$

$$\text{Motor capacity } a_x < M(v), \tag{2b}$$

$$\text{Traction circle } (\frac{a_y}{a_{\max,y}})^2 + (\frac{a_y}{a_{\max,y}})^2 < 1, \tag{2c}$$

$$\text{Collision } |n^i - n^j| > n_{\min}, |s^i - s^j| > s_{\min}, \tag{2d}$$

where $M(v), a_{\max,y}, a_{\max,x}$ are determined by system identification.

We encode these constraints as quadratic barrier costs in the agent cost function as follows

$$\begin{aligned} J^i(x_0, u) = & \\ & \sum_{k=0}^{K}\{-s^i_{k+1} + \sum_{j\neq i}\alpha^i s^j_{k+1} + x^{\mathrm{T}}_{k+1}Q^i x_{k+1} + u^{i\mathrm{T}}_k R^i u_k\} \\ & + J_{\text{col}}(x^+, \alpha^i) + J_{\text{bdry}}(x^+) + J_{\text{ctrl}}(u^i), \end{aligned} \tag{3a}$$

$$\begin{aligned} J_{\text{col}}(x, \alpha^i) = & \sum_{j\neq i}\mathbf{1}\{|\Delta s^{ij}| < s_{\min} \,\&\, |\Delta n^{ij}| < n_{\min}\} \\ & C_{\text{col}}(\alpha^i)((|\Delta s| - s_{\min})^2 + (|\Delta n| - n_{\min})^2), \\ & \Delta s^{ij} = s^i - s^j, \Delta n^{ij} = n^i - n^j, \end{aligned} \tag{3b}$$

$$\begin{aligned} J_{\text{bdry}}(x) = & \mathbf{1}\{n_i > \rho_L(s)\}C_{\text{bdry}}(n_i - \rho_L(s))^2 + \\ & \mathbf{1}\{-n_i > \rho_R(s)\}C_{\text{bdry}}(-n_i - \rho_R(s))^2, \end{aligned} \tag{3c}$$

$$J_{\text{ctrl}}(u_i) = \mathbf{1}\{u_i \notin U_i\}C_u \inf_{\bar{u}\in U_i}\|u_i - \bar{u}\|_2^2, \tag{3d}$$

where $\mathbf{1}\{\cdot\}$ evaluates to unity when the condition is true, otherwise it is zero.

B. Model Predictive Path Integral (MPPI)

Model Predictive Path Integral [18] is a sampling-based model predictive control method that can solve optimal control problems with nonlinear dynamics and cost functions. The original MPPI and its derivatives have been tested on the AutoRally vehicle platform [11] as a capable controller for single-vehicle racing.

C. Iterated Best Response (MPPI-IBR)

We combine MPPI with iterated best response (IBR)[7] to create a game-theoretic version of MPPI. IBR converts the multi-agent game to multiple single agent optimal control problems by iteratively solving the optimal control problem for each agent while modeling other agents as unreactive. While there is no guarantee of convergence, if IBR converges for a specific problem, the solution furnishes a Nash equilibrium.

D. Iterated Linear Quadratic Regulator (iLQR)

iLQR approximates a nonlinear problem around a reference trajectory with linear dynamics and linear quadratic cost function. The approximated problem can be solved backward in time by maintaining a linear quadratic value function. The

solution can be used to update the reference trajectory until convergence.

E. iLQGame

iLQGame [19] is a game-theoretic version of iLQR. In the backward pass, iLQGame solves the Nash policy of a linear quadratic game [20]in each stage. iLQGame can find an open or closed loop local Nash policy [21] for all agents for the approximated game.

V. Results

We demonstrate the BuzzRacer platform and evaluate the performance of the MPPI, MPPI-IBR, iLQR, and iLQGame in a series of simulations and experiments. The simulations are run on a computer equipped with an Intel i9-10920X CPU operating at 3.5GHz and NVIDIA RTX3090 GPU. The sampling procedure for MPPI and MPPI-IBR is run in parallel on GPU with CUDA.

A. Simulation

For comparing two controllers, two identical simulation vehicles are spawned in random positions on the race track, each controlled by a candidate controller. To promote overtaking behavior, the two agents start close to each other and the vehicle in the rear is given an initial speed advantage. The experiment concludes when either vehicle finishes two complete laps. Then, the initial states of the two vehicles are swapped and the experiment is repeated. This removes any potential bias in the randomness of initial starting states. Due to the tight radius and narrow width of the grid-style tracks, overtaking in identical cars is challenging, and we designed three wider, smoother tracks (Figure 6) that enable the agents to better interact. The benchmark experiments are conducted with two iterations (when applicable) with a horizon of 20 time steps at 0.02s per step with aggressiveness $\alpha = 1$. MPPI, MPPI-IBR uses 1024 samples in rollout.

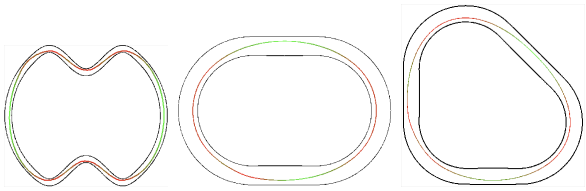

Fig. 6. Tracks used in simulation.

MPPI and MPPI-IBR utilize the GPU for rollout of sampled control sequences while iLQR and iLQGame run purely on the CPU. Figure 7 shows the maximum controller frequency achieved for MPPI-IBR and iLQGame. These results are achieved on a desktop computer equipped with an Intel i9-10920X 3.5GHz CPU and an Nvidia RTX3090 GPU. All controllers can run in real-time with satisfactory performance in some configurations.

Table II shows the comparison between the two base controllers, namely, MPPI and iLQR, with MPPI showing an advantage in all tracks. MPPI admits nonlinear dynamics without the need for approximation like iLQR, and the control noise sampling reduces its sensitivity to initial guesses. These benefits likely contributed to the performance edge.

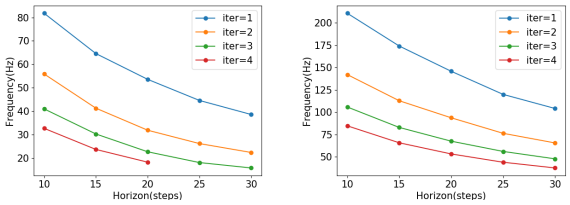

Fig. 7. Time Performance of iLQGame and MPPI-IBR.

TABLE II
MPPI vs iLQR*

Track	Win Ratio (as follower)	Win Ratio (as leader)	Win Ratio (overall)
nascar	0.55	0.66	0.61
triangle	0.50	0.70	0.60
sine	0.78	0.62	0.70

*The win ratio indicates the winning ratio of the first controller.
**Results averaged over 100 simulations

TABLE III
MPPI-IBR vs MPPI

Track	Win Ratio (as follower)	Win Ratio (as leader)	Win Ratio (overall)
nascar	0.54	0.54	0.54
triangle	0.52	0.74	0.63
sine	0.46	0.54	0.50

TABLE IV
iLQGame vs iLQR

Track	Win Ratio (as follower)	Win Ratio (as leader)	Win Ratio (overall)
nascar	0.54	0.86	0.70
triangle	0.64	0.86	0.75
sine	0.39	0.80	0.60

Tables III and IV show the benchmark result of MPPI-IBR and iLQGame against their non-game-theoretic counterparts. While they both performed better than their respective baseline, iLQGame demonstrates a clear advantage. iLQGame models the game dynamics in an LQGame in each step, while MPPI-IBR is ignorant of agent interaction. Agent interaction manifests only as updates to the predictions of opponent agents. The benchmark experiments of MPPI-IBR and iLQGame shown in Table V indicate a more formal treatment of the dynamic game as in iLQGame pays dividends in performance in a two-car racing game.

B. Experiment

We tested BuzzRacer as a platform for multi-agent racing and validated the real-life performance of the tested controllers in a series of two-car races on our grid-style track.

To further improve real-time performance, we used the planned trajectory from each controller as a reference trajectory for a Stanley tracking controller [22]. This allows

TABLE V
MPPI-IBR VS ILQGAME

Track	Win Ratio (as follower)	Win Ratio (as leader)	Win Ratio (overall)
nascar	0.44	0.30	0.37
triangle	0.24	0.08	0.16
sine	0.10	0.17	0.14

us to run the tracking controller asynchronously at a higher frequency and reduces latency in the control pipeline.

Since BuzzRacer cars are identical, the controllers achieve very close lap times, and rarely build enough speed differential for overtaking. In simulations, we manually imposed an initial speed differential to promote overtaking. In experiments, we restricted the performance of one agent to achieve similar results.

We found that the tested controllers were able to navigate the track without collision, and utilized tactics similar to faking and pushing as seen in human drivers. Figure 8 shows snapshots of an overtake. Videos of the experiments are available on the BuzzRacer project website [14].

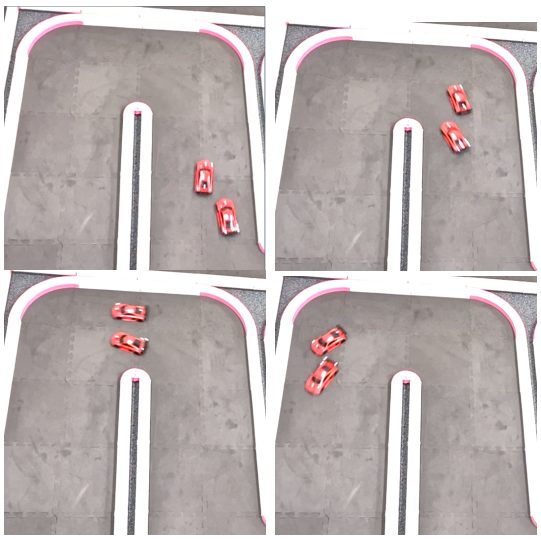

Fig. 8. Snapshots of an overtake

VI. CONCLUSIONS

In this work, we demonstrated a novel miniature autonomous racing platform BuzzRacer that costs less than $200 for a single unit. It is suitable for developing and testing multi-agent planning and control algorithms. The modular software stack minimizes the overhead of transitioning from simulation to experiment. With this platform, we proposed a formulation of the two-car racing game and compared the performance of two state-of-art game-theoretic controllers and their non-game theoretic base versions in a series of simulation and on-car experiments. Current work focuses on the elimination of the rather expensive Optitrack system using on-board time-of-flight sensors for relative localization and collision avoidance.

REFERENCES

[1] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 458–488, 2022.

[2] V. A. Laurense, J. Y. Goh, and J. C. Gerdes, "Path-tracking for autonomous vehicles at the limit of friction," in *American Control Conference*, Seattle,WA, 2017, pp. 5586–5591.

[3] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars," in *European Control Conf.*, Limassol, Cyprus, Jun. 2018, pp. 1341–1348.

[4] J. L. Vázquez, M. Brühlmeier, A. Liniger, A. Rupenyan, and J. Lygeros, "Optimization-based hierarchical motion planning for autonomous racing," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Paris, France, 2020, pp. 2397–2403.

[5] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[6] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Duerr, "Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4257–4264, Jul. 2021.

[7] Z. Wang, T. Taubner, and M. Schwager, "Multi-agent sensitivity enhanced iterative best response: A real-time game theoretic planner for drone racing in 3D environments," *Robotics and Autonomous Systems*, vol. 125, p. 103410, Mar. 2020.

[8] S. L. Cleac'h, M. Schwager, and Z. Manchester, "ALGAMES: A fast solver for constrained dynamic games," in *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation, Jul 2020.

[9] M. Prajapat, K. Azizzadenesheli, A. Liniger, Y. Yue, and A. Anandkumar, "Competitive policy optimization," Online, 2021.

[10] F. Camara and C. Fox, "Unfreezing autonomous vehicles with game theory, proxemics, and trust," *Frontiers in Computer Science*, vol. 4, 2022.

[11] B. Goldfain, P. Drews, C. You, M. Barulic, O. Velev, P. Tsiotras, and J. M. Rehg, "Autorally: An open platform for aggressive autonomous driving," *IEEE Control Systems Magazine*, vol. 39, no. 1, pp. 26–55, 2019.

[12] "MIT Racecar." [Online]. Available: https://racecar.mit.edu/

[13] 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.

[14] "Buzzracer project website." [Online]. Available: https://github.com/DCSLgatech/BuzzRacer

[15] A. Liburdi, "Development of a scale vehicle dynamics test bed," Electronic Theses and Dissertations, University of Windsor, 2010. [Online]. Available: https://scholar.uwindsor.ca/etd/195

[16] C. You and P. Tsiotras, "Vehicle modeling and parameter estimation using adaptive limited memory joint-state UKF," in *American Control Conference*, Seattle,WA, 2017, pp. 322–327.

[17] X. Qian, A. de La Fortelle, and F. Moutarde, "A hierarchical model predictive control framework for on-road formation control of autonomous vehicles," in *IEEE Intelligent Vehicles Symposium*, 2016, pp. 376–381.

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

[19] D. Fridovich-Keil, V. Rubies-Royo, and C. J. Tomlin, "An iterative quadratic method for general-sum differential games with feedback linearizable dynamics," in *IEEE International Conf. on Robotics and Automation*, 2020, pp. 2216–2222.

[20] T. Basar, *Dynamic Noncooperative Game Theory, 2nd Edition*. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611971132.fm

[21] M. Rowold, "A preview of open-loop and feedback nash trajectories in racing scenarios," 2023. [Online]. Available: arXiv:2402.01918[cs.RO]

[22] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *American Control Conference*, New York, NY, 2007, pp. 2296–2301.