Dynamic Multi-Team Racing: Competitive Driving on 1/10-th Scale Vehicles via Learning in Simulation

Anonymous Author(s)
Affiliation
Address
email

Abstract: Autonomous racing is a challenging task that requires vehicle handling at the dynamic limits of friction. While single-agent scenarios like Time Trials are solved competitively with classical model-based or model-free feedback control, multi-agent wheel-to-wheel racing poses several challenges including planning over unknown opponent intentions as well as negotiating interactions under dynamic constraints. We propose to address these challenges via a learning-based approach that effectively combines model-based techniques, massively parallel simulation, and self-play reinforcement learning to enable zero-shot sim-to-real transfer of highly dynamic policies. We deploy our algorithm in wheel-to-wheel multi-agent races on scale hardware to demonstrate the efficacy of our approach. Several instances of these hardware races are provided in a supplementary video.

Keywords: Multi-Agent RL, Autonomous Racing, Sim-to-Real Transfer

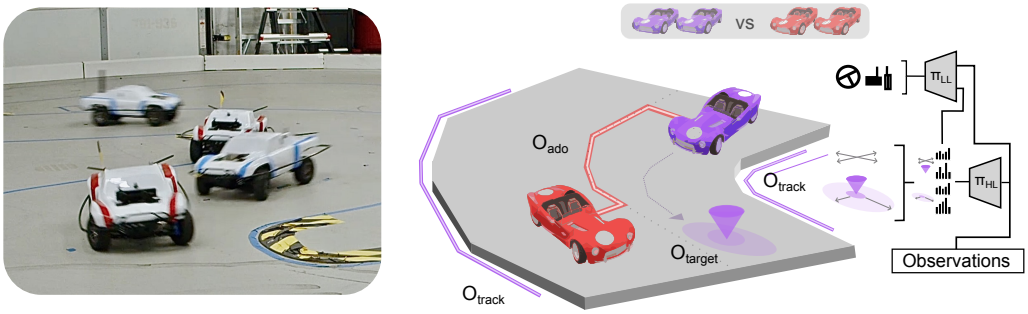

Figure 1: We train multi-team racing agents in simulation and transfer them directly to hardware. Our approach leverages a hierarchical policy as a layer of abstraction between strategic planning and continuous control. A high-level categorical policy predicts goal distributions for a low-level policy to achieve, effectively projecting long-term decision making into a limited preview horizon.

1 Introduction

Autonomous racing in a multi-vehicle team setting is a highly challenging task. It requires safely handling a vehicle at its dynamic limits while simultaneously reasoning about opponents and team strategy. These challenges push the limits of autonomous navigation and have, thus, attracted considerable academic interest. At the same time, solving autonomous racing promises to impact decision-making for real-world autonomous vehicles. The frequent interactions and driving at the limits of controllability are reminiscent of certain near-accident scenarios which are absent from common autonomous vehicle research datasets [1].

The multi-agent learning community has focused largely on simulated challenges such as [2, 3, 4]. We see multi-team racing as a step towards a simulation and hardware benchmark that is both

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

accessible and realizable without the use of proprietary hardware, due to the vibrant open-source community revolving around scaled-car racing [5].

In the considered multi-team racing task, two teams of two agents compete for a team win. A team's rank is determined by its best-ranked team member at the end of a race. Solving this task effectively is challenging because it requires both long-term strategic planning (e.g. coordination with teammates or setting up overtaking maneuvers over multiple corners of the track) and fast, responsive short-term planning for swift and precise handling of the vehicle (e.g. to avoid collisions or to counteract unexpected slippage).

Most autonomous racing systems follow a highly structured pipeline similar to traditional driving pipelines. This allows for fine-grained design of each component within perception, planning, and control. In contrast, purely learning-based approaches to autonomous racing allow addressing fundamental questions of embodied AI research such as the emergence of social behavior, embodied learning setup, and safe real-world policy transfer. However, these have mostly been tested in simulated environments, due to the difficulty of large-scale experience data collection on real hardware. Simulations present many ways to diversify and extend data collection. In particular, simulating competitive multi-agent scenarios enables self-play with large agent populations and avoids potential damage to hardware systems. Yet to this day, sim-to-real transfer in domains where neither driving at the limits nor wheel-to-wheel contact dynamics are well modeled remains a challenge.

In this work, we demonstrate a sim-to-real approach to multi-team racing. We expand on successful hierarchical policy architectures and leverage parallelized self-play in simulation to learn competitive agents with a variety of emergent behaviors, that transfer onto real hardware. Toward this aim, our work makes the following contributions:

- A light-weight GPU-accelerated simulation environment for multi-team racing, augmenting first principles-based with data-driven models to facilitate highly parallelized learning.
- A hierarchical policy structure that serves as a layer of abstraction to disentangle long-term team-centric strategic planning from short-term ego-centric continuous control.
- Training in simulation with zero-shot transfer to scale race cars, demonstrating dynamic maneuvers in highly interactive wheel-to-wheel multi-team racing scenarios on hardware.

Specifically, we review related work in Section 2, followed by our approach for environment modeling and control in Sections 3 and 4, respectively. We describe details of our race car platform in Section 3.1, followed by results of transfer for wheel-to-wheel racing on hardware in Section 5.

2 Related Work

This work presents a platform that advances motion planning for autonomous racing, multi-agent reinforcement learning (MARL) applications, and practical approaches for dynamics modeling and sim-to-real transfer. Pushing the limits of autonomous vehicles through autonomous racing has seen a recent surge of interest [6]. Investigated systems include both reduced scale platforms [5, 7, 8], full scale platforms [9], as well as arcade-style simulated systems [10, 11, 12].

Motion Planning Motion planners can be grouped into three main categories. First, obstacle avoidance planners treat ado-agents as moving obstacles without explicit consideration of interactions between agents and their objectives. This includes collision-free path identification via dynamic programming [13], filter-based prediction [14], reinforcement learning (RL) [15], graph-based planning [16], and adapting a low-level optimization either via learned reward specifications [17] or trajectory warm-starts [18]. Second, game-theoretic planners explicitly represent interaction strategies of ado agents and iterate until an equilibrium is reached. This includes formulations as sequential bi-matrix games [19] or iterative dynamic programming in belief space [20], with analysis commonly limited to 1-vs-1 scenarios. Iterative best response for 6 vehicles is showcased in [21], with the prevalent drawback that game theoretic planners introduce a trade-off between solution quality and real-time capabilities. Finally, learning-based planners only implicitly reason about

interactions from prior data and do not compute explicit ado plans. In [22], hand-crafted scenarios and rewards are combined with model-free RL to beat human players in the game of Grand Turismo, while [11] learns a latent world-model that reasons about opponent behavior for competitive 1-vs-1 racing via self-play. The hierarchical approach in [12] enables a high-level policy to propose lane change sequences to a low-level controller in response to ado behavior. Our approach leverages a hierarchical policy structure as a layer of abstraction between team-centric strategic planning and ego-centric continuous control, learning efficient multi-agent interactions from experience and displaying emergent behavior via self-play. This further enables real-time inference when scaling to many-vs-many races on hardware.

Multi-Agent Reinforcement Learning This work considers a mixed, cooperative-competitive, racing scenario, where teams receive a joint reward for their overall performance, with only very limited regularization terms signaling individual contribution. Similar problems are extensively studied in the MARL literature [23, 24, 25]. In particular, we consider the centralized training decentralized execution (CTDE) paradigm with factored (action) value functions as described in [26, 27, 28]. These approaches attempt to learn individual payoff assignments between cooperating agents. This enables decentralized execution by greedy action with respect to the individual payoff. Alternatively factored policies can be directly trained using actor-critic approaches such as Proximal Policy Optimization [29]. We make use of CTDE to train the high-level policy with DecSARSA, an on-policy adaptation of decoupled Q-Networks [28], learning credit assignment for team members.

Dynamics Modeling and Sim-to-Real Transfer To accelerate learning, we leverage massively parallel simulation and training running directly on the GPU [30]. A wide variety of dynamics models are used in practice for autonomous vehicles. For example, the authors in [31] use a dynamic single-track bicycle model with Pacejka tire models [32] and model randomization for policy learning. The same model formulation was used in [33] containing system identification of parameters for the F1Tenth platform. In [34], physics-informed basis functions are combined with Bayesian regression to fit a dynamics model used for predictive control. For a comprehensive overview of dynamics models for racing, see [6]. Finally, our work uses simulation-to-reality (sim-to-real) transfer. See [35, 36] for select surveys on established sim-to-real approaches.

3 Simulation and Dynamics Modeling

Our approach leverages a highly parallelized, light-weight simulator for time-efficient behavior learning on GPUs. The simulator merges data-driven dynamics obtained through system identification on hardware with simplified analytic models to generate high-fidelity interactive scenarios.

3.1 Hardware Platform

We employ the TRIKart hardware platform developed by the Toyota Research Institute [8], a variation of the F1Tenth vehicle design [5]. Each vehicle has an Nvidia Jetson onboard computer to handle communication with the VESC motor driver, steering servo, and the base station computer. The base station provides real-time control commands together with state estimates obtained from an OptiTrack motion capture system. We further outfit each vehicle with a Lexan cover augmented with ABS bars to mitigate damage during vehicle-to-vehicle contact scenarios.

3.2 Dynamics Model

The simulator leverages a switched dynamics model conditioned on the contact state of the system. Out of collision, we employ the kinematic bicycle model [37] augmented with data-driven models. During collisions, we employ the dynamic bicycle model [13]. Both variations are implemented as elementary tensor operations in PyTorch for rapid data generation [38], and described below.

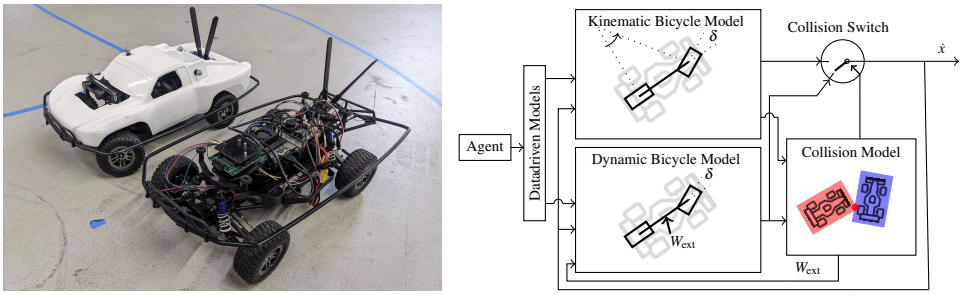

Figure 2: *Left* – 10th-scale TRIKart platform. *Right* – Block diagram of the switched dynamics. The simulator augments a kinematic bicycle model with data driven front slip angle and drive train dynamics learned from the real vehicle. If a collision event is detected, the dynamics are switched temporarily to a lower fidelity dynamic model that directly models the effects of external wrenches.

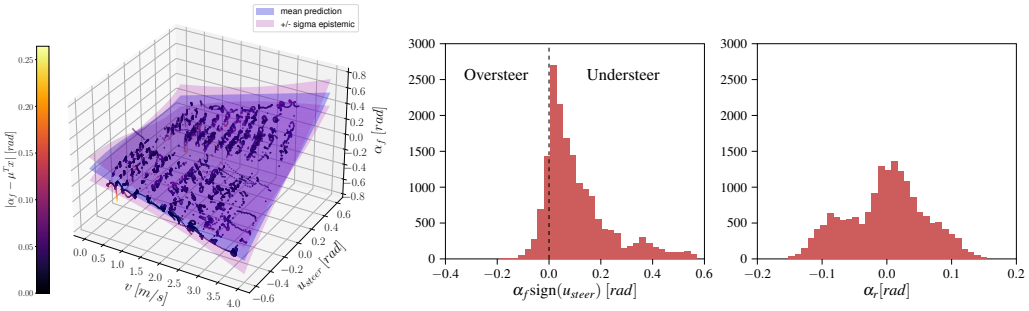

Figure 3: A linear model with odd steering angle features is fit on hardware data using BLR to capture model mismatches based on understeer. *Left* – Model predictions with epistemic uncertainty. *Right* – Data of front, α_f, and rear slip angle, α_r. The front slip angle is scaled such that positive values correspond to understeer at the front axle, which covers around 87% of the recorded data.

Out-of-contact model: We combine the kinematic bicycle model from [37] with feature-based predictors of the front tire slip angle as well as a data-driven model of the longitudinal dynamics (see Section 3.3). The dynamics of the kinematic model are then given by

$$\dot{x} = f_{\text{kin}}(x,u) = \begin{bmatrix} v\cos(\theta+\beta) \\ v\sin(\theta+\beta) \\ \frac{v\tan(u_{\text{steer}}+\alpha_f)}{\sqrt{L^2+(l_r\tan(u_{\text{steer}}+\alpha_f))^2}} \\ \Phi_{\text{long}}(v,u_{\text{steer}},u_{\text{acc}}), \end{bmatrix}, \tag{1}$$

where $x = [X,Y,\theta,v]$ is the vehicle state, $u = [u_{\text{steer}},u_{\text{acc}}]$ are the controls, the center of mass slip angle is $\beta = \arctan\left(\tan(u_{\text{steer}}+\alpha_f)\frac{l_{\text{r}}}{l_{\text{r}}+l_{\text{f}}}\right)$, α_f is the predicted front slip angle, and Φ_{long} represents the longitudinal acceleration model. The data-driven augmentation yields a sufficiently high-fidelity representation of the hardware dynamics, but does not explicitly model reaction forces. We identify collisions based on rectangular hit boxes and simulate contact wrenches with a spring model accounting for penetration depth.

In-contact model: When a collision event is detected, the simulation switches to the dynamic bicycle model from [13, 33]. The resulting contact wrenches are then explicitly propagated into the state dynamics, switching back to the kinematic model after a pre-specified cool-down or if the rear slip angle falls below a threshold. A schematic of the switched dynamics is provided in Figure 2.

3.3 System Identification

The parameter identification and data-driven model estimation proceed directly on the hardware system. The steering commands are calibrated based on trajectories recorded with a motion capture

system under the condition of minimal tire slip, allowing for the steering angle to be inferred from the center of mass slip angle. While the resulting kinematic bicycle model provides a good initial approximation of the dynamics, it does not capture higher-order effects relating to e.g., variable drive-train acceleration or tire traction. To better model these effects, we collected a dataset of trajectories exciting the full range of throttle and steering inputs, while avoiding the oversteer regime due to the vehicles' tendency to understeer when running at the dynamic limits on our track. See Figure 3. A longitudinal acceleration model is then trained to minimize the one-step prediction error based on velocity, steering, and throttle inputs. To model lateral deviations at higher speeds we estimate the slip angle of the front tire via Bayesian linear regression (BLR). Hereby we use simple hand crafted features that are odd in the steering angle to fit our data. We assume our measured slip stems from a linear model in the features with i.i.d. Gaussian measurement noise $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ and a spherical Gaussian prior on the model weights $\varepsilon_w \sim \mathcal{N}(0, \sigma_p^2 \mathbb{I})$. During training the dynamics are randomized by drawing slip models from the posterior distribution and injecting noise into the received commands. See Appendix A and B.2 for details.

4 Agent Structure and Training Process

We define multi-team racing as a finite-horizon decentralized MDP (decMDP) [39, 40] (brief review in Appendix C). We leverage a GPU-accelerated implementation of the dynamics described in the previous section to train competitive multi-team racing agents. Our approach combines coarse high-level planning with fine-grained low-level control within a hierarchical policy structure trained via bilevel optimization. An overview of the bilevel learning procedure is illustrated in Figure 4.

4.1 Hierarchical Control Structure

Competitive control in interactive settings requires both long-term strategic decision-making as well as short-term responsive action selection. These objectives occur at different layers of abstraction, with low-level control conditioned on high-level decisions, thus lending themselves to bilevel optimization procedures [41, 42, 43, 44]. Here, we leverage a hierarchical policy design [45, 46] to effectively guide ego-centric low-level decision-making through team-centric high-level goals. To this end, we train a discrete high-level policy to suggest goal states, and a low-level policy to resolve local velocity and heading commands in order to achieve these goals, as illustrated in Figure 1.

High-level control: Our formulation enables the high-level policy to propose goal states to the low-level controller. This introduces a layer of abstraction that helps to disentangle long-term strategic planning from short-term control. Goal states are encoded by their relative position along the track as well as their associated uncertainty. The uncertainty encodes goal importance with respect to the high-level strategy and enables the low-level policy to weight short-term gains against long-term considerations. We further leverage coarse high-level representations via categorical distributions to encode mode-switching behavior, e.g., *"slow down and keep to the right"*. This encourages diverse high-level strategies and can provide favorable exploration characteristics [47]. The high-level policy is obtained as an ε-greedy evaluation of a state action value function trained with Decoupled SARSA, an on-policy variation of Decoupled Q-Networks [28]. This leverages value decomposition across team members and enables Centralized Training with Decentralized Execution (CTDE) [26, 48]. We further adopt Munchausen Value Iteration [49] for regularization.

Low-level control: The low-level controller receives high-level goals with associated time budgets, predicting steering and velocity commands. This effectively augments short-horizon local reasoning with long-horizon strategic planning. However, the goal states only act as soft constraints, enabling the low-level controller to retain flexibility and prioritize local decision-making when necessary. The low-level policy is trained via PPO [50] under the primary objective of clearing high-level goals, while considering behavior smoothness, obstacle avoidance, and local track geometry. The strategic team decisions are thereby implicitly provided through high-level goals.

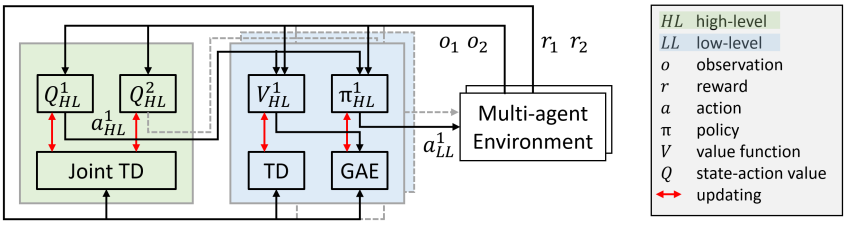

Figure 4: The hierarchical policy structure abstracts team-centric strategy from ego-centric control. The high-level predicts discrete goal distributions and is trained jointly across team member leveraging value decomposition with DecSARSA. The low-level learns continuous actions to satisfy individual goal states with PPO. Training proceeds in parallel simulation via bilevel optimization.

Hardware-level control: The steering and velocity commands sampled from the low-level policy serve as references to a hardware-level PD tracking controller. The PD controller generates steering and acceleration commands, running at a faster rate to mitigate the effects of unmodeled dynamics.

4.2 Training in Simulation

The hierarchical policy is trained via bilevel optimization in parallel simulation. Training consists of two phases, learning basic driving skills during pre-training and effective racing during fine-tuning.

Pre-training: Initially, the low-level policy will have to learn rudimentary driving skills that enable goal-reaching behavior. During this stage, the agent will only generate low quality learning signals for the high-level policy that may induce representational collapse [51]. We therefore pre-train the low-level policy for 500 iterations with PPO on goals sampled from the initially uniform high-level policy, competing against Pure Pursuit opponents to simulate capable ado behavior.

Fine-tuning: The bilevel optimization then alternates every 50 iterations. High-level training leverages DecSARSA to refine strategic behavior via goal prediction, while low-level training employs PPO to align local controls with the adapted high-level goal distribution. During this stage, we sample opponents as a mixture from prior policy checkpoints to encourage competitive behavior to emerge through self-play, and opponents steered by pure-pursuit control to prevent strategic collapse [52]. We consider both joint and independent high-level Q-functions for fine-tuning, referring to the two approaches as RL Team and RL Independent.

Observations and rewards: The high-level policy observes ego state, rank information, and track boundaries over a preview horizon. It further receives noisy ado odometry data within a view-limited neighborhood, which is processed by a multi-head attention encoder to extract ado relevance with respect to the current ego state into a latent representation [53]. The low-level observations additionally contain the high-level goals, as in Figure 1. The high-level policy is primarily rewarded based on team rank information, encouraging strategic behavior that improves positioning of the ego team, and receives zero-sum overtaking reward and a small term encouraging ego progress. The low-level policy is primarily rewarded for accomplishing high-level goals at their respective timings, with regularization objectives that modulate vehicle collisions, off-track driving, and control smoothness.

5 Experiments

We evaluate the efficacy of our approach by first training policies in simulation and then transferring them directly to the scale race car hardware. Policy learning is conducted on a single Nvidia RTX 2080Ti with 2048 parallel environments taking roughly 8 hours. Policy inference is conducted on the CPU for all experiments with the high-level policy running at 0.5 Hz and the low-level policy running at 20 Hz. The employed learning parameters and training details are summarized in Appendix B.

The results from fine-tuning in simulation are summarized in Figure 5. We evaluate winrates of the RL Team agent over the course of training against itself at convergence, a variation that uses inde-

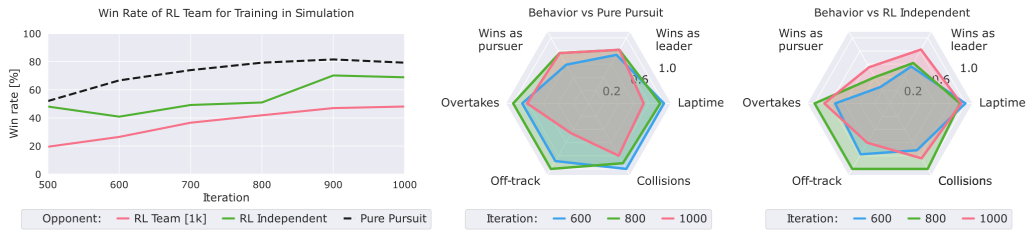

Figure 5: Evaluation in simulation shows consistent improvement of winrate and driving skill over the course of training. We compare our RL Team agent to an egoistic variation with independent high-level learners and a pure pursuit baseline, achieving a winrate of 79% and 69%, respectively.

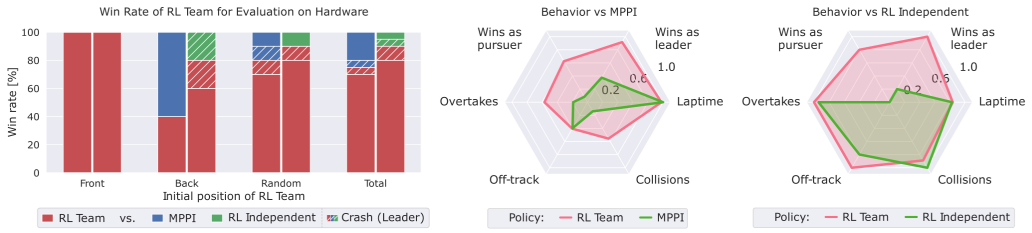

Figure 6: Transfer to hardware shows strong performance of our RL Team agent compared to the RL Independent and MPPI baselines, achieving winrates above 80% and 70%, respectively. We observe consistent wins when starting in front and a strong ability to win from the back of the starting grid. Stripes indicate crash scenarios in which a pursuing agent did not finish the race, colored by winner.

pendent learning, and the pure pursuit baseline. We further quantify behavior characteristics based on races against the latter two agents. The winrate of RL Team steadily improves to reach around 80% against pure pursuit and 70% against RL Independent. In particular, winrate from behind approaches 60% against RL Independent, while laptime and time spent off-track are reduced. The trend in other factors is less clear-cut, as e.g. improved ability to block will increase collision count while reducing the need to overtake later on. Generally, we find high-level training to significantly improve performance, yield better positioning on the track, and enable smoother driving.

We further evaluate our approach against two model predictive path integral control (MPPI) [54] agents, running collision avoidance based on an MPPI motion predictions for all 3 ado cars (see [55]), and against two RL Independent agents on hardware for 20 five-lap races each. The 20 races are split into 3 starting configurations: *Front* – RL Team starting in 1st and 2nd (for 5 races), *Back* – RL Team starting in 3rd and 4th place (for 5 races), and *Random* – randomized starting configurations (for 10 races). The maximum velocity is set to 3.5 $[\frac{m}{s}]$ for all hardware experiments. The results from the hardware evaluation are summarized in Figure 6. On hardware, we observed single-agent average laptimes across 10 laps of 6.8s for RL Team, 7.6s for RL Independent and 6.7s for MPPI . We found that RL Team was great at keeping its position once in front, for example by using the rear team member to fend off opponents, resulting in a 91% winrate against MPPI and a 100% winrate against RL Independent, when starting from the front. We further observe that RL Independent agents display a more reckless driving style reflected in the number of collisions, crashes, and offtrack time, see Figure 6, which forces the RL Team agents to negotiate an increased number of (near-)collision events as reflected by the behavioral differences in the radar charts. We refer the reader to Appendix D for the un-normailzed data.

Emergent behavior observed in the simulator and on hardware is showcased in Figure 7. The hierarchical policies learn multi-car overtaking maneuvers, altruistic blocking where the rear team-member significantly slows down and attempts to block attacking opponents, and pitting maneuvers to jostle opponents out of the way, yielding successful sim-to-real transfer of competitive behaviors.

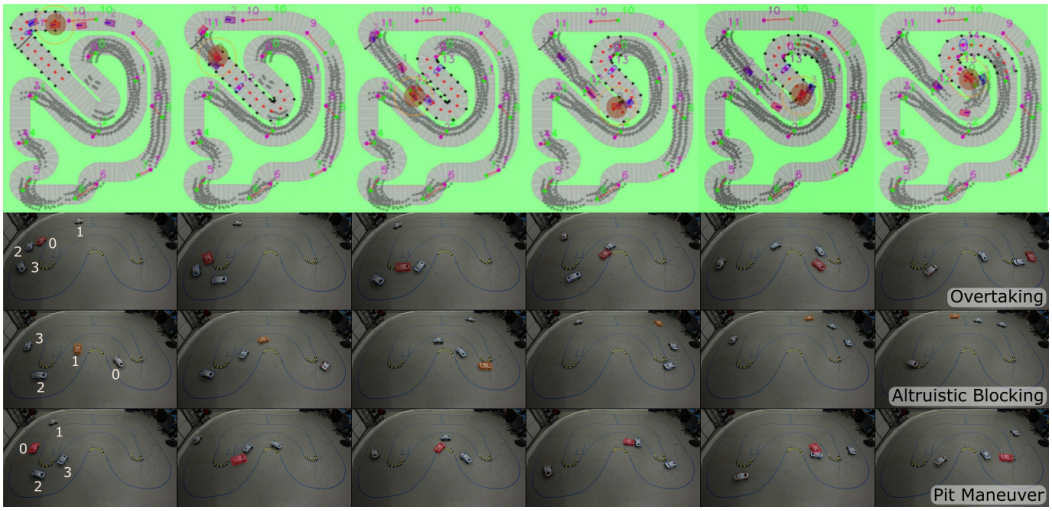

Figure 7: *Top* – GPU-accelerated simulation, visualizing high-level targets and track observations. Agent 0 veers to the right track edge (frame 3), cuts the corner (frame 4), pushes agent 3 off-track (frame 5), and takes the lead (frame 6). *Bottom* –Three observed emergent behaviors. *Overtaking*: Agent 0 performs a double overtake on two ado agents. *Altrusitic Blocking*: Agent 1 stays back to block agents 2 and 3, allowing for agent 0 to pull away. *Pit Maneuver*: Agent 1 pulls up behind agent 3 and performs a pitting maneuver. See also supplementary material for video sequences.

6 Limitations

We briefly discuss limitations to our current approach, which point towards avenues for future work. **Ado observation sensitivity**: Agents retain sensitivity to their ado observations even when strategic interactions are not likely. This was more pronounced during hardware experiments when coupled with system delays and the sim-to-real gap of the dynamics model, sometimes causing exaggerated deviations from the race line. **Collisions between teammates and corner cutting**: Teammates occasionally jostle for positions, and trade penalties for potential position gains by cutting corners, which could be refined by adapting the problem formulation. **Dynamic limits**: While this work brings the vehicles to the limits of friction and demands corrections for understeer, the presented pipeline does not allow for agents to act at the torque limits of the scaled cars. Future work could investigate this dynamics regime while exploring methods to further reduce hardware system delays.

7 Conclusion

We propose a novel light-weight simulator for learning dynamic multi-team wheel-to-wheel racing for zero-shot sim-to-real transfer. The simulator combines classical analytic models with data-driven components and is GPU-accelerated to enable massively parallel simulation. We further introduce a hierarchical control structure that disentangles high-level strategic planning from low-level continuous control. The high-level policy suggests goals to the low-level controller and is represented with categorical variables to encode mode-switching behavior. The low-level policy predicts continuous controls and incorporates high-level guidance into its local decision making, but retains the ability to deviate if required. The high-level policy is obtained from centralized training across team members and therefore provides strategic information beyond single-agent control.

We train multi-agent racing policies in simulation and deploy onto the TRIKart scale race car platform without further adaptation. The transferred policies retain their efficacy and are able generate dynamic behavior on hardware when tested with a team of MPPI ado agents, including overtaking, blocking, and even pit maneuvers. Future work will be directed at further reducing the sim-to-real gap, while tackling hardware system engineering challenges in supporting more ado agents. Generally, our results indicate great promise for combining parallelized simulation with hierarchical policy abstraction for achieving zero-shot sim-to-real transfer of dynamic multi-team racing behaviors.

References

[1] M. Zhou, J. Luo, J. Villella, Y. Yang, D. Rusu, J. Miao, W. Zhang, M. Alban, I. Fadakar, Z. Chen, et al. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. *arXiv preprint arXiv:2010.09776*, 2020.

[2] M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[3] K. Kurach, A. Raichuk, P. Stanczyk, M. Zajkac, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4501–4510, 2020.

[4] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869*, 2020.

[5] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam. F1tenth: An open-source evaluation environment for continuous control and reinforcement learning. *Proceedings of Machine Learning Research*, 123, 2020.

[6] 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 Journal of Intelligent Transportation Systems*, 3:458–488, 2022.

[7] O. So, P. Drews, T. Balch, V. Dimitrov, G. Rosman, and E. A. Theodorou. Mpogames: Efficient multimodal partially observable dynamic games, 2022.

[8] V. Dimitrov, P. Drews, T. Balch, X. Cui, A. A. Allaban, G. Rosman, and S. G. McGil. Trikart: Enabling human-centered and rapid design of shared control in adas. In *H-MRS workshop*, 2022.

[9] J. Kabzan, M. de la Iglesia Valls, V. Reijgwart, H. F. C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, and R. Siegwart. AMZ driverless: The full autonomous racing system. May 2019.

[10] T. Seyde, W. Schwarting, S. Karaman, and D. Rus. Learning to plan optimistically: Uncertainty-guided deep exploration via latent model ensembles. *arXiv preprint:2010.14641*, 2020.

[11] W. Schwarting, T. Seyde, I. Gilitschenski, L. Liebenwein, R. Sander, S. Karaman, and D. Rus. Deep latent competition: Learning to race using visual control policies in latent space. *arXiv preprint arXiv:2102.09812*, 2021.

[12] R. S. Thakkar, A. S. Samyal, D. Fridovich-Keil, Z. Xu, and U. Topcu. Hierarchical control for cooperative teams in competitive autonomous racing. *arXiv preprint arXiv:2204.13070*, 2022.

[13] A. Liniger, A. Domahidi, and M. Morari. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, 36(5):628–647, sep 2015. ISSN 01432087.

[14] A. Raji, A. Liniger, A. Giove, A. Toschi, N. Musiu, D. Morra, M. Verucchi, D. Caporale, and M. Bertogna. Motion planning and control for multi vehicle autonomous racing at high speeds. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2775–2782. IEEE, 2022.

[15] B. Evans, H. A. Engelbrecht, and H. W. Jordaan. Learning the subsystem of local planning for autonomous racing. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 601–606. IEEE, 2021.

[16] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp. Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3149–3154. IEEE, 2019.

[17] R. Reiter, J. Hoffmann, J. Boedecker, and M. Diehl. A hierarchical approach for strategic motion planning in autonomous racing. *arXiv preprint arXiv:2212.01607*, 2022.

[18] S. He, J. Zeng, and K. Sreenath. Autonomous racing with multiple vehicles using a parallelized optimization with safety guarantee using control barrier functions. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3444–3451. IEEE, 2022.

[19] A. Liniger and J. Lygeros. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology*, 28(3):884–897, 2019.

[20] W. Schwarting, A. Pierson, S. Karaman, and D. Rus. Stochastic dynamic games in belief space. *IEEE Transactions on Robotics*, 37(6):2157–2172, 2021.

[21] 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*, 125:103410, 2020.

[22] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, et al. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

[23] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[24] K. Zhang, Z. Yang, and T. Başar. Multi-Agent reinforcement learning: A selective overview of theories and algorithms. Nov. 2019.

[25] S. Gronauer and K. Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pages 1–49, 2022.

[26] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[27] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020.

[28] T. Seyde, P. Werner, W. Schwarting, I. Gilitschenski, M. Riedmiller, D. Rus, and M. Wulfmeier. Solving continuous control via q-learning. *International Conference on Learning Representations*, 2023.

[29] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu. The surprising effectiveness of PPO in cooperative Multi-Agent games. Oct. 2022.

[30] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[31] E. Chisari, A. Liniger, A. Rupenyan, L. Van Gool, and J. Lygeros. Learning from simulation, racing in reality. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8046–8052. IEEE, 2021.

[32] H. B. Pacejka and E. Bakker. The magic formula tyre model. *Vehicle system dynamics*, 21 (S1):1–18, 1992.

[33] D. Zahrádka. Optimization-based control of the f1/10 autonomous racing car. *Czech Technical University in Prague, Master's Thesis*, 2020. URL https://people.ciirc.cvut.cz/~klapajar/studenti/F3-DP-2020-Zahradka-David-thesis.pdf.

[34] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[35] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-Real transfer in deep reinforcement learning for robotics: a survey. Sept. 2020.

[36] X. Hu, S. Li, T. Huang, B. Tang, and L. Chen. Sim2real and digital twins in autonomous driving: A survey. May 2023.

[37] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium (IV)*, pages 1094–1099. IEEE, 2015.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[39] F. A. Oliehoek and C. Amato. *A Concise Introduction to Decentralized POMDPs*. Springer International Publishing.

[40] A. Beynier, F. Charpillet, D. Szer, and A.-I. Mouaddib. DEC-MDP/POMDP. In *Markov Decision Processes in Artificial Intelligence*, pages 277–318. John Wiley & Sons, Inc., Hoboken, NJ USA, Mar. 2013.

[41] H. v. Stackelberg et al. Theory of the market economy. 1952.

[42] J. Bracken and J. T. McGill. Mathematical programs with optimization problems in the constraints. *Operations research*, 21(1):37–44, 1973.

[43] A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2): 276–295, 2017.

[44] T. Seyde, J. Carius, R. Grandia, F. Farshidian, and M. Hutter. Locomotion planning through a hybrid bayesian trajectory optimization. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5544–5550. IEEE, 2019.

[45] M. Wulfmeier, A. Abdolmaleki, R. Hafner, J. T. Springenberg, M. Neunert, T. Hertweck, T. Lampe, N. Siegel, N. Heess, and M. Riedmiller. Compositional transfer in hierarchical reinforcement learning. *arXiv preprint arXiv:1906.11228*, 2019.

[46] T. Seyde, W. Schwarting, I. Gilitschenski, M. Wulfmeier, and D. Rus. Strength through diversity: Robust behavior learning via mixture policies. In *Conference on Robot Learning*, pages 1144–1155. PMLR, 2022.

[47] T. Seyde, I. Gilitschenski, W. Schwarting, B. Stellato, M. Riedmiller, M. Wulfmeier, and D. Rus. Is bang-bang control all you need? solving continuous control with bernoulli policies. *Advances in Neural Information Processing Systems*, 34:27209–27221, 2021.

[48] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[49] N. Vieillard, O. Pietquin, and M. Geist. Munchausen reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4235–4246, 2020.

[50] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[51] E. Nikishin, M. Schwarzer, P. D'Oro, P.-L. Bacon, and A. Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, pages 16828–16847. PMLR, 2022.

[52] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

[53] S. Iqbal and F. Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International conference on machine learning*, pages 2961–2970. PMLR, 2019.

[54] G. Williams, A. Aldrich, and E. Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

[55] 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 International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[56] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

A System Identification Details

In order to identify the slip model we use Bayesian Linear regression. Analogously to [56], Chapter 3, we compute the posterior and inference distributions at $\hat{x}$ as

$$p(w_{\text{slip}}|X,Y) = \mathcal{N}(w_{\text{slip}};\mu,\Sigma) \tag{2}$$

$$\mu = \left(X^T X + \frac{\sigma_n^2}{\sigma_p^2}\mathbb{I}\right)^{-1} X^T Y \tag{3}$$

$$\Sigma = \left(\frac{1}{\sigma_n^2}X^T X + \frac{1}{\sigma_p^2}\mathbb{I}\right)^{-1} \tag{4}$$

$$p(\hat{y}|\hat{x},X,Y) = \mathcal{N}(\hat{y};\mu^T\hat{x},\ \underbrace{\hat{x}^T\Sigma\hat{x}}_{\text{epistemic}} + \underbrace{\sigma_n^2}_{\text{aleatoric}}) \tag{5}$$

where the resulting uncertainty splits into an epistemic and an aleatoric component. In order to fit the model we use the following features and targets,

$$X = \begin{bmatrix} | & | & | \\ u_{\text{steer}} & v u_{\text{steer}} & u_{\text{steer}}^3 \\ | & | & | \end{bmatrix} \quad Y = \begin{bmatrix} | \\ \alpha_f \\ | \end{bmatrix} \tag{6}$$

along with hand-tuned noise parameters in Table 1.

σ_p^2 [rad]	σ_n^2 [rad]
1.0	0.01

Table 1: Employed BLR noise parameters.

B Learning Parameters and Domain Randomization

We elaborate on aspects of the training setup below, discussing the reward functions and domain randomization used. Throughout, we consider high-level steps that consist of $T_{LL} = 20$ low-level steps. At each high-level step, a goal state is sampled and then held fixed for the duration of the low-level substeps.

B.1 Reward functions

We provide details on the reward function used to train the high-level (HL) and low-level (LL) policies. For ease of notation, we introduce the following variables: λ^T and λ^E represent team and ego rank, respectively, p and $\dot{p}$ denote the vehicle position and velocity in track frame, respectively, μ^g and Σ^g correspond to the mean and standard deviation of the multivariate goal state distributions, respectively, w_i is an indicator for whether tire i is off-track, a represents the actions, and ξ is a collision indicator. The wheel-on-track term features a slight asymmetry that up-weights penalty terms when either the front or rear axle are off-track, while the collision term up-weights rear-ending collisions - we omitted these features from the reward equations for visual clarity. We further introduce index t denoting time and t_{LL} as the fraction of time left until the next high-level step.

$$r_t^{HL} = c_1(\lambda_t^T == 0) + c_2(\lambda_0^T - \lambda_t^T) + c_3\frac{\lambda_{t-1}^T - \lambda_t^T}{1+\min(\lambda_{t-1}^T - \lambda_t^T)} + c_4 e^{-\lambda_t^T} + c_5 e^{-\lambda_t^E} \tag{7}$$

$$r_t^{LL} = c_6\frac{1}{2\pi|\Sigma_t^g|^{1/2}}e^{-\frac{1}{2}(p_t-\mu_t^g)^T\Sigma_t^{g^{-1}}(p_t-\mu_t^g)+\frac{1}{4}\dot{p}_t} + c_7\sum_{i=0}^{3}w_t^i + c_8\Delta p_t + c_9 e^{-|\dot{p}|} + c_{10}(a_t - a_{t-1})^2 + c_{11}\xi_t \tag{8}$$

	Symbol	Value	Term explanation
HL	c_1	4.0e-1	Team is in first place
	c_2	5.0e-2	Team improved over starting rank
	c_3	1.0e-1	Zero-sum overtaking reward
	c_4	1.0e-3	Regularize with team rank
	c_5	1.0e-2	Regularize with ego rank
LL	c_6	$\mathbb{I}(t_{LL} < 0.1)$ * (1.0e+0)	Clear goal at end of low-level episode
	c_7	-1.0e-3	Penalize wheels off track
	c_8	$\mathbb{I}(\Delta p_t > 2.5)$ * (-2.5e-1)	Penalize large progress jumps (cutting)
	c_9	-2e-3	Encourage high velocity
	c_{10}	-6e-4	Regularize action smoothness
	c_{11}	-5e-3	Penalize collisions

Table 2: Reward function components

B.2 Domain Randomization

To improve the sim-to-real transfer of the learned policies we randomize the dynamics simulation during training time. The simulated slip is randomized by sampling a new weight vector from the posterior distribution in Equation 2 at the beginning of every training episode. We further randomize select model parameters with additive noise δ summarized in Table 3.

Symbol	Distribution	Explanation
$v_{\max}$	$\mathscr{U}(3.5, 4.5)[\frac{m}{s}]$	Additive noise velocity cap
δ_{l_f}	$\mathscr{U}(-0.02, 0.02)[m]$	Additive noise to front length
δ_{l_r}	$\mathscr{U}(-0.02, 0.02)[m]$	Additive noise to rear bicycle length
$\delta_{u_{\text{steer}}}$	$\mathscr{U}(-0.03, 0.03)[rad]$	Additive noise to all steering commands
$\delta_{u_{\text{acc}}}$	$\mathscr{U}(-0.15, 0.05)[-]$	Additive noise to simulated throttle commands

Table 3: Ramdomized model parameters

In order to improve robustness of the policies we add uniform random noise, drawn independently from $\mathscr{U}(-0.005, 0.005)$, to all observations except the rank and team rank observations.

Note: Upon publication we will open source the simulator and code to train policies.

C Decentralized Markov Decision Process Formulation

We formulate multi-team racing as a finite horizon decentralized Markov Decision Process (DecMDP), that is characterised by the tuple $(\gamma, S, T, A, \Omega, O, R)$, where γ is the discounting factor, S is the set of joint states, T is the probability densitiy fucntion $p(s'|s,a)$ of transitioning from s to s', when agents take the joint action a, A is the set of action sets available to all agents, Ω is the set of all observation spaces, O is the set of all observation functions, and R is the set of reward functions across agents, where each reward function returns the reward for taking joint action a at state s.

In practice, we define multi-team racing in a symmetric fashion across agents such that their action spaces are identical. The observation functions are defined as outlined in Section 4, such that each agent only observes their velocity, nearby track boundaries, number of wheels that are off track, rank, remaining time in the episode, last actions, and respective ado agents.

D Hardware Experiments

The radar plots in Figure 6 are intended to give the reader an intuitive understanding of the behavior of the various agents. For clarity the axis are normalized with the maximum value of the respective statistic aggregated during both the races against of RL Team (RL T.) against MPPI agents and RL Independent (RL Ind.). In this appendix the raw data is listed in Table 4, along with brief explanations. All statistics are averaged over the 20 respective races. As a guideline, the average race duration is around 40s.

Statistic	RL T. vs MPPI	RL T. vs RL Ind.	Explanation
Laptime [s]	(8.44 , 8.62)	(8.36, 8.26)	Avg. Laptime of fastest team member
Wins as leader	(0.92, 0.38)	(1.00, 0.2)	Win rate w/ team member starting in 1st
Wins as pursuer	(0.63, 0.08)	(0.80, 0.00)	Win rate w/ opponent starting in 1st
Overtakes	(2.85, 0.95)	(5.00, 4.70)	Number of overtakes per team
Off-track [s]	(1.57, 1.54)	(3.89, 3.104)	Time off track per team
Collisions	(1.0 , 0.25)	(1.6, 1.8)	number of collisions per team member

Table 4: Un-normalized data from the radar charts in Figure 6.