TC-Driver: Trajectory Conditioned Driving for Robust Autonomous Racing - A Reinforcement Learning Approach

Edoardo Ghignone[1], Nicolas Baumann[1], Mike Boss[2] and Michele Magno[1]

***Abstract*—Autonomous racing is becoming popular for academic and industry researchers as a test for general autonomous driving by pushing perception, planning, and control algorithms to their limits. While traditional control methods such as Model Predictive Control (MPC) are capable of generating an optimal control sequence at the edge of the vehicles' physical controllability, these methods are sensitive to the accuracy of the modeling parameters. This paper presents TC-Driver, a Reinforcement Learning (RL) approach for robust control in autonomous racing. In particular, the TC-Driver agent is conditioned by a trajectory generated by any arbitrary traditional high-level planner. The proposed TC-Driver addresses the tire parameter modeling inaccuracies by exploiting the heuristic nature of RL while leveraging the reliability of traditional planning methods in a hierarchical control structure. We train the agent under varying tire conditions, allowing it to generalize to different model parameters, aiming to increase the racing capabilities of the system in practice. The proposed RL method outperforms a non-learning-based MPC with a 2.7 lower crash ratio in a model mismatch setting, underlining robustness to parameter discrepancies. In addition, the average RL inference duration is 0.25 ms compared to the average MPC solving time of 11.5 ms, yielding a nearly 40-fold speedup, allowing for complex control deployment in computationally constrained devices. Lastly, we show that the frequently utilized end-to-end RL architecture, as a control policy directly learned from sensory input, is not well suited to model mismatch robustness nor track generalization. Our realistic simulations show that TC-Driver achieves a 6.7 and 3-fold lower crash ratio under model mismatch and track generalization settings, while simultaneously achieving lower lap times than an end-to-end approach, demonstrating the viability of TC-driver to robust autonomous racing.**

I. INTRODUCTION

Autonomous car racing pushes the boundaries of algorithmic design and implementation in perception, planning, and control [1]. Thus, it is a valuable asset for researchers to push the limits of autonomous driving [2, 3]. This offers many benefits, such as enhancing road safety, reducing carbon emissions, transporting the mobility-impaired, and reducing driving-related stress [4, 5]. Therefore, autonomous car racing serves as a catalyst for the long-term goals of general autonomous vehicles [6].

In recent years, many autonomous racing competitions have emerged, held at prestigious robotics conferences, and received considerable attention in the fields of robotics and Artificial Intelligence (AI). Among other competitions, the F1TENTH racing platform is gaining popularity and attracting researchers from all over the globe. F1TENTH is a semi-regular autonomous racing competition involving a race car on a scale of 1:10. The competition offers both a simulator environment [7] and a physical racing platform. As the standardized platform offers little room for improvement on the hardware side, the main challenges are raised on the algorithmic side [7]. Namely, the control layer becomes the key focus of development, as the system in itself is highly nonlinear and the behavior of the car must be taken into consideration at the edge of stability [6].

Current State-of-the-Art (SotA) racing controllers utilize optimal control methods such as MPC [1, 6, 2, 3]. While MPC can guarantee optimality of the planned trajectory and tracking within its receding horizon, it heavily relies on the accuracy of the modeling parameters. Particularly in the context of autonomous car racing, the model inaccuracies of the lateral tire forces are critical for high-performance racing. These forces are notoriously difficult to model, and the tires' behavior is highly nonlinear [8], making modeling errors potentially very dangerous. In real racing scenarios, a tire modeling mismatch is very likely to occur, as high wear and tear and changes in weight modify the initial parameters [8]. While there exist several previous works that have attempted to address this issue using learning-based methods [9, 10] for MPC, we assess and highlight the feasibility and performance of an RL approach.

RL methods offer a Machine Learning (ML)-based solution to handle these mismatches. Said kind of techniques proved to be able to handle high-performance racing on different occasions [11, 12, 13, 14]. The mentioned architectures are end-to-end learned, meaning that they learn the optimal control policy directly from sensory input. Further, these works do not focus on the generalisation capabilities on model mismatch and only show partial [12] or poor [13] generalisation results to unseen tracks.

We propose a hybrid architecture that specifically addresses the robustness towards model mismatch and track generalization, inspired by the two-layer planner-controller separation that is often present in robotic systems [15, 1]. In this framework, the planning layer (e.g., Frenet planner [16]) is responsible for generating a safe and performant trajectory, while the control layer is dedicated to generating control inputs in order to make the system follow the given trajectory. According to this layout, we consider the planner to be given, and use a RL agent for the low-level control, exploiting the learning capabilities of such models to heuristically handle model mismatch and track generalization, and leverage the safety and reliability of traditional planning methods [16]. Namely, we present a trajectory-conditioned RL controller (TC-Driver) that is able to generalize to different tracks as its task is to track an arbitrary trajectory.

Furthermore, we train the RL agent under constantly varying tire parameters, such that it learns to generalize to varying

[1] Associated with Center for Project Based Learning, D-ITET, ETH Zurich
[2] Associated with D-INFK, ETH Zurich

model conditions, allowing lap completion in a racing setting. We show that such a hierarchical structure is beneficial to the learning objective of a RL agent when compared to an end-to-end setting based on previous SotA [13, 17, 12, 11], as later shown in Section III.

The proposed architecture offers multiple benefits:

- **Robustness to Modeling Mismatch:** The proposed controller adapts to parameter mismatches due to deep learning's generalization capabilities and RL's heuristic properties. In particular, we focus on the notoriously difficult to model lateral tire forces required for high-speed racing [6, 9, 10, 8]. It is shown in Table I that the proposed method yields better model mismatch robustness compared to the non-learning-based MPC and the end-to-end setting. The MPC, under model mismatch, demonstrates a crash ratio of 38.1% as opposed to the TC-Driver with 14.29%, resulting in a factor of ~ 2.7 lower crash ratio. Further, the end-to-end architecture demonstrates a crash ratio of 95.24%, a lower crash ratio of TC-Driver is demonstrated by the factor of $\sim$ 6.7. Thus underlining the model mismatch robustness.
- **Track Generalization Capabilities:** The proposed architecture can better generalize to unseen tracks as the observation given to the RL model has no general reference to the track itself but only a partial trajectory. The proposed architecture yields superior generalization capabilities on unforeseen tracks when compared to the end-to-end setting based on previous SotA [13, 11, 17]. As shown in Table II the average crash ratio of the end-to-end architecture is 38.09%, while the TC-Driver's crash ratio is 12.7%, demonstrating a lower crash ratio by the factor of 3. When compared to MPC, our method does not show better results. However, on two tracks out of three, it yields the same performance as the optimal control method.
- **Computational Benefit:** The envisioned control structure is beneficial in terms of computational load compared to a full MPC setting. To quantify, the RL inference has an average duration of $0.25\,\mathrm{ms}$ compared to the average MPC solving time of $11.5\,\mathrm{ms}$, as in Table III.

II. METHODOLOGY

The RL terminology follows the convention of [18]. The main goal of our architecture is to train an agent operating a race car that is aware of a given trajectory under the influence of noise applied to the tire friction coefficients. That is, in every episode, the environment will have different tire modeling parameters. Thus, the agent learns to handle the tire parameter modeling mismatch as it generalizes on how to handle these conditions during training, ultimately allowing for robust tracking of a given trajectory.

A. Simulation Environment

The F1TENTH simulation environment [7] aims to offer an OpenAI gym compatible wrapper [19]. Within the environment, the vehicle's dynamics are modeled with the *Single Track* model [20], which is also known as the *Bicycle Model*, to realistically simulate Ackermann-steered vehicles.

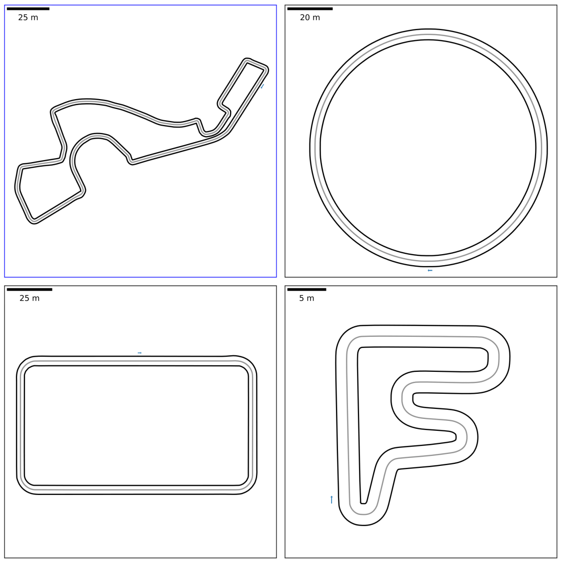

Figure 1: Training track *SOCHI* depicted in blue. Testing tracks *Circle*, *Square* and *F*, which are unseen during training. The tracks vary in length from $470\,\mathrm{m}$ to $89\,\mathrm{m}$ in length, while the gray line depicts the centerline. Track width varies from $5\,\mathrm{m}$ to $3\,\mathrm{m}$, for reference the width of the car is $0.31\,\mathrm{m}$.

μ, $C_{S,f}$, $C_{S,r}$ model the friction, the cornering stiffness on the front axle, and the cornering stiffness on the rear axle, respectively as in [21, 20]. The F1TENTH environment has been modified to be able to inject noise into the parameters, allowing the investigation of robustness in terms of tire modeling inaccuracies. The simulation environment offers the following car's dynamic state: $s_{dyn} = [s_x,\, s_y,\, \psi,\, v_x,\, v_y,\, \dot{\psi}] =$ [global x position, global y position, yaw angle with respect to the positive x-axis, longitudinal velocity, lateral velocity, yaw rate].

Further, the simulation environment provides sensory input in the form of a LiDAR scan made of 1080 points over 270° coverage area around the car. To summarize, the observation of the environment is $obs_{gym} = [scan,\, s_x,\, s_y,\, \psi,\, v_x,\, v_y,\, \dot{\psi}]$

The action space of the gym environment solely consists of continuous actions $a = [v, \delta]$, where v is the desired longitudinal velocity and δ is the steering angle of the agent.

The reward function is defined in Eq. (1) inspired by [11, 13].

$$r_t = \begin{cases} -c & \text{if track constraints are violated} \\ p_{t+1} - p_t & \text{otherwise} \end{cases} \tag{1}$$

where $c = 0.01$ and $p_{t+1} - p_t$ is the track advancement within one simulation time step. The range of the reward is calculated as $p_{t+1} - p_t \in [-T_s V_{max}, T_s V_{max}]$ and evaluates to $p_{t+1} - p_t \in [0, 0.1]$ with the simulation time step $T_s = 10\,ms$ and $V_{max} = 10\frac{m}{s}$. The track constraints are defined as $|n_t| \geq \frac{1}{2} w_{track} - \frac{3}{2} w_{car}$. If the car deviates from the center line by more than half of the track's width w_{track} minus a safety margin based on $\frac{3}{2}$ times the car's width w_{car}, it violates the track constraints. This emulates on the RL the spatial soft constraints also employed on the MPC as defined in [6].

B. Reinforcement Learning Architectures

In this section, we introduce both the frequently used end-to-end RL architecture [11, 14, 13, 12] and the low-level RL trajectory tracker, with their underlying architecture, environment interaction, and hyperparameters. To ensure a fair comparison, both settings use the same underlying model-free architecture, namely Soft Actor Critic (SAC) [22] featuring off-policy learning, entropy regularization, and double learning. It is derived from the Stable Baselines 3 (SB3) [23] implementation.

1) End-to-End Racer: To generate a baseline for comparisons, we utilized the frequently used model-free end-to-end architecture of [11, 17, 13], possessing a slightly modified observation space with respect to the one previously defined in Section II-A. The chosen observation space recasts the observation in a *Frenet frame*, which is a representation relative to a trajectory, as in [11, 17, 13]. The new observation is $obs_{end2end} = [p, n, \psi, v_x, v_y, \dot{\psi}]$ = [progress along the path, perpendicular deviation from the path, yaw relative to the trajectory, relative heading, longitudinal velocity, lateral velocity, yaw rate].

The agent learns a control policy with online environment interaction based on the previously defined reward function in Section II-A. Since advancement-based rewards like ours were broadly tested [13, 11, 12, 17], we consider this agent a reasonable comparison model.

2) Trajectory Conditioned Driver: The proposed low-level trajectory tracker (TC-Driver) tracks the spatial trajectory generated by a high-level planner. Within this work, a pre-generated Model Predictive Contouring Controller (MPCC) trajectory is used, which has been custom implemented for this task following [6]. That is, the track has already been traversed by a MPC, and the logged trajectory can then be used by subsequent RL agents. It is worth mentioning that this trajectory could be chosen arbitrarily, such as, for example, by using the center-line trajectory instead of the time-optimal MPC trajectory.

The observation space of the proposed low-level trajectory tracker is altered compared with the end-to-end setting. To enable trajectory following, we add a sample of the optimal trajectory relative to the current position of the car. This sample consists of 30 points, rotated and translated to be in the car's frame of reference.

Therefore, the observation space in the hierarchical setting is newly defined as $obs_{traj} = [traj, p, n, \psi, v_x, v_y, \dot{psi}]$ = [relative trajectory, progress along the path, perpendicular deviation from the path, relative heading, longitudinal velocity, lateral velocity, yaw rate].

Furthermore, the reward function is slightly modified to penalize the agent for excessive swerving from the trajectory:

$$r_t = \begin{cases} -c & \text{if constraints violated} \\ p_{t+1} - p_t - |n_t| & \text{otherwise} \end{cases} \quad (2)$$

where $|n_t|$ is the spatial deviation perpendicular to the target trajectory of the planner and the constraint violations are the same as in Eq. (1).

C. Tire Parameter Randomization

The F1TENTH simulation environment utilizes the single-track dynamic model of [20]. To apply randomness to the tire coefficients, we added Gaussian noise at each reset of the gym environment during training. The noise was centered at the nominal value friction, used in the MPC to find the optimal trajectory.

To find the standard deviation, the limit of tire friction at which MPC would not be able to correctly complete a lap was analyzed. Then the standard deviation of the noise was set to be half of that value for the noise to be mostly (but not entirely) inside the range of values that allow MPC to finish a lap. The numerical values are: $\mu_{noisy} \sim \mathcal{N}(1.0489, 0.0375)$.

D. Implementation

The used environment is based on an adapted version of the F1TENTH gym racing environment [7]. Both RL agents were implemented using SB3 SAC algorithm. SAC was initialized with gamma at 0.99, an episode length of 10000, batch size of 64, train frequency of 1, and using the Multilayer Perceptron (MLP) policy.

III. RESULTS

In this section, we evaluate the proposed trajectory tracking agent against the end-to-end agent as well as the MPC method with and without the tire parameter randomization during training. Evaluation metrics consist of lap-time, the ability to handle different track conditions, and the ability to drive unseen tracks. In simulation, one can use the optimal trajectory of a MPC with exact model parameters (without *tire noise*) as the ground truth reference. For the results presented in Section III-A and Section III-B, an agent was trained for $1e6$ timesteps for each algorithm on the track named SOCHI, which can be seen in Fig. 1.

A. Comparison of End-to-End and TC-Driver

It is to be mentioned that a direct comparison between the end-to-end and the trajectory tracker based on the reward itself is not possible as both settings have different reward functions and observation spaces. As a result, we compare the traversed distances of the proposed trajectory tracking agent and the end-to-end agent to that of a MPC with exact modeling and simulation parameter matching. Fig. 2 shows that the end-to-end agent swerves much more compared to the trajectory tracker that mostly tracks the MPC generated trajectory in a stable manner. Analysing the total run, the end-to-end agent chooses a trajectory that totals 470.22 m in length, which is significantly longer than the proposed TC-Driver trajectory;

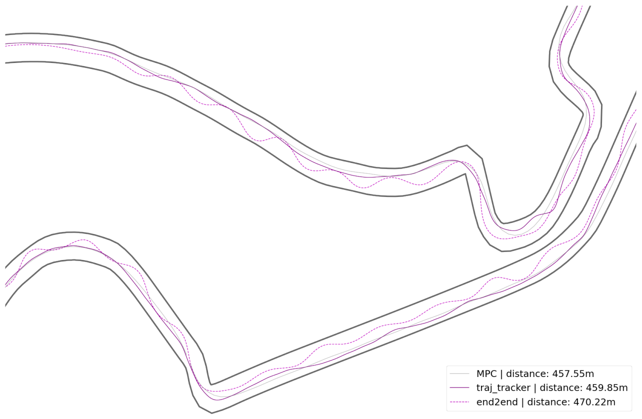

Figure 2: Comparison of the end-to-end and trajectory tracking agent with the optimal trajectory of a MPC.

namely, 12.67 m compared to the 2.3 m of excessive trajectory length of the trajectory tracking agent. This behavior clearly does not follow a realistic purpose as the end-to-end agent swerves a lot on the straights for which the optimal trajectory given by the MPC is close to completely straight.

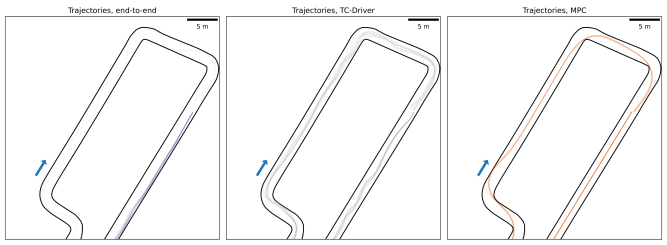

Figure 3: Agents that were trained under tire friction randomization within the MPC tolerance for 21 runs, tested in an environment outside the trained tire friction domain. Left is the end-to-end agent; middle shows the proposed trajectory tracking agent; right shows MPC with a crash rate of 38.1% due to the high parameter mismatch.

B. Robustness to Tire modeling Mismatch

To test the capabilities of the algorithms to generalize to different tire friction, 21 randomly extracted values were utilized during test laps. To better test the generalization capabilities, these friction parameters were extracted in an interval that was predominantly outside the training range. In detail, the normal distribution had a mean 0.2 lower than the nominal one, with the same standard deviation as in the training phase, i.e., 0.0375. The MPC was run with the nominal system model, i.e., the tires' friction was not changed, to simulate model mismatch. The three different models were run on the track, starting from the same position, for one lap.

In Fig. 3 one can see a trajectory extract, with the 21 laps superimposed one on the other. Due to the parameters mismatch, the MPC does not manage to finish the lap in 38.1% of the times, as shown in Table I. Yet, it still achieves the best and most consistent lap time of $46.261\,\mathrm{s}$ with $0.054\,\mathrm{s}$ standard deviation. While the trajectory conditioned RL driver is significantly slower with an average lap time of $53.528\,\mathrm{s}$, it has the lowest crash percentage of 14.29%. Compared to the end-to-end RL driver with a crash ratio of 95.24% (only a single run was completed successfully), the proposed trajectory conditioned algorithm performs notably better when faced with model mismatch.

Investigating the crash ratio of Table I, the proposed TC-Driver outperforms the end-to-end architecture by a factor of ~ 6.7 and the MPC by a factor of ~ 2.7. In this setting, the proposed method shows better generalization capabilities to model mismatch. Regarding the MPC it has to be said that such a result is expected, as the tire mismatch lies outside of the modeling domain. A solution for such a situation would be the integration of learnable parameters within the MPC model, as in [10]. Thus, this result does not exhibit superiority to MPC, but rather demonstrates a case in which RL can be utilized in the mitigation of model mismatch.

Table I: Lap time results of 21 runs, comparison with imperfect knowledge of dynamics on the training track. Average lap time t_μ in seconds (lower is better); Standard deviation of the lap times t_σ (lower is better); Percentage of crashes during the runs (lower is better).

	Lap time t_μ [s]	Lap time t_σ [s]	Crashes
MPC	**46.261**	**0.054**	38.10%
end-to-end	59.030	n.a.	95.24%
TC-Driver	53.528	0.348	**14.29** %

C. Track Generalization Capabilities

To test the trajectory conditioned driver's ability to generalize to race tracks beyond the training track of *SOCHI*, it was evaluated on three additional unforeseen tracks, namely *Circle*, *Square* and *F* track visible in Fig. 1.

The agents were started at 21 different positions along these tracks and drove a single lap each. To further emphasize the generalization capabilities to arbitrary trajectories, the trajectories used for conditioning the TC-Driver were the centerlines of the test tracks, instead of the time-optimal raceline of the MPCC as in the training phase.

Table II depicts the described runs on the unseen tracks. The MPC clearly outperforms both RL methods, as expected. It shows the fastest lap times on all test tracks with the lowest standard deviation, while never crashing. Averaging the lap time standard deviation and comparing them between the end-to-end and TC-Driver, yields a factor ~ 3 smaller deviation in favor of the TC-Driver. Thus, the TC-Driver manages to complete the laps in a significantly more consistent time. Further, evaluating the average crash ratio of the end-to-end (38.09%) and TC-Driver (12.7%) results in a factor of 3 lower crash ratio for the TC-Driver. Therefore, the TC-Driver shows better generalization capabilities on unseen tracks.

Interestingly, however, is the inferior crash ratio of the TC-Driver on the *F*-track. The suspected reason for this result is the scale of the track with respect to the training track *SOCHI* and the other test tracks *Circle* and *Square*.

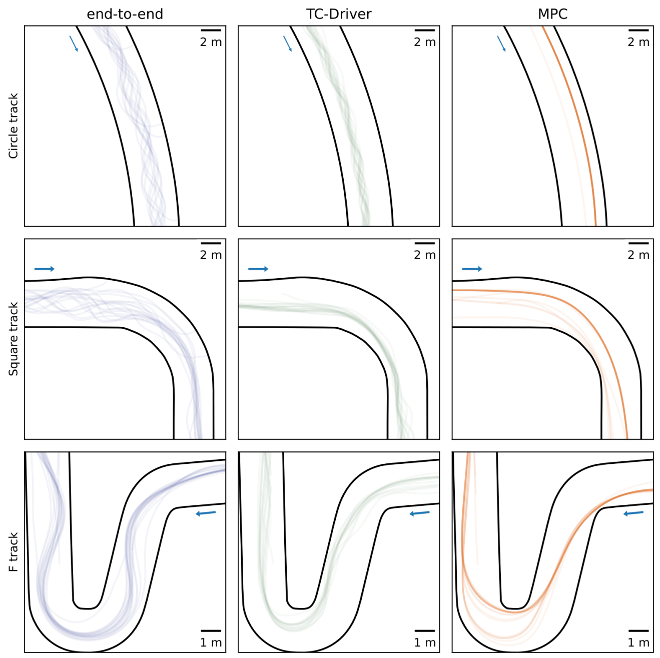

Figure 4: Generalization runs of the end-to-end, TC-driver and MPC algorithms from left to right, on the unseen test tracks *Circle*, *Square* and *F* respectively, from top to bottom. Consisting of 21 runs without tire friction randomization.

Table II: Lap time results of 21 runs on the unseen tracks *Circle*, *Square* and *F* with zero model mismatch. Average lap time in seconds (lower is better); Standard deviation of the lap times (lower is better); Percentage of crashes during the runs (lower is better).

Track	Driver	Lap time t_μ [s]	Lap time t_σ [s]	Crashes
Circle	MPC	**33.111**	**0.129**	0.00%
	end-to-end	39.436	0.456	61.90%
	TC-Driver	39.548	0.335	**0.00 %**
Square	MPC	**39.544**	**0.251**	**0.00%**
	end-to-end	49.107	0.605	28.57%
	TC-Driver	46.992	0.333	**0.00%**
F	MPC	**10.582**	**0.210**	**0.00%**
	end-to-end	14.303	1.750	23.81%
	TC-Driver	11.753	0.274	38.10%

D. Computation Time

Lastly, we focus on the computational time of the utilized control methods. Table III depicts the average computation time of each method and their respective standard deviation. The MPC's average computation duration is approximately 11 ms with a rather high standard deviation of 0.9 ms. The reason for the higher deviation arises from the nature of quadratic programming, which is subject to constantly varying solving conditions. On the other hand, both RL algorithms show a significantly lower and more constant inference time of approximately 0.26 ms. Thus, the RL computation time is faster by a factor of roughly 40.

Table III: Average computation time of the utilised control methods and their respective standard deviation.

	Computation Time t_μ [ms]	Computation Time t_σ [ms]
MPC	11.2	0.9
end-to-end	**0.26**	0.05
TC-Driver	0.27	**0.04**

IV. CONCLUSION

We presented TC-Driver, a hierarchical approach to autonomous racing, using RL to track trajectories generated by a traditional high-level planner. Given imperfect modeling of parameters, MPC's optimality does not hold, leading to slower lap times and potentially even crashes. RL offers a viable approach to this solution by generalizing to different driving conditions. Yet, end-to-end RL methods, rely on states that are not fit for efficient generalization to different tracks and model mismatch. Combining a traditionally generated trajectory as an observation for a RL agent tracking the trajectory under changing conditions alleviates these shortcomings. We evaluated and compared these approaches in the simulated F1TENTH autonomous racing environment [7]. The proposed TC-Driver shows that it can adapt to model mismatch scenarios that the non-learning based MPC, based on [6] fails to handle. Further, it outperforms the model-free end-to-end architectures based on [11, 13, 17], in all metrics regarding the robustness to model mismatch. It achieves lower and more consistent lap times, compared to the end-to-end agent, and has the lowest overall crash ratio in the model mismatch setting.

Future work on this topic is the deployment and comparison of the TC-Driver on the physical F1TENTH system. This allows us to further investigate and evaluate the model mismatch robustness as well as the sim-to-real capabilities. Lastly, a highly interesting RL approach would be the utilization of a model-based RL architecture, as inspired by [12].

The code for reproducing all mentioned RL and MPCC F1TENTH implementations is available at: https://github.com/ETH-PBL/TC-Driver.

ACKNOWLEDGMENT

We would like to thank Dr. Christian Vogt, Dr. Andrea Carron and Dr. Alexander Liniger of ETH Zürich, for their constructive and fruitful algorithmic discussions.

REFERENCES

[1] J. Kabzan, M. de la Iglesia Valls, V. Reijgwart, H. F. C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. B. 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," *CoRR*, vol. abs/1905.05150, 2019. [Online]. Available: http://arxiv.org/abs/1905.05150

[2] C. K. Law, D. Dalal, and S. Shearrow, "Robust model predictive control for autonomous vehicles/self driving cars," 2018. [Online]. Available: https://arxiv.org/abs/1805.08551

[3] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: a predictive control approach," 2019. [Online]. Available: https://arxiv.org/abs/1901.08184

[4] T. J. Crayton and B. M. Meier, "Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy," *Journal of Transport & Health*, vol. 6, pp. 245–252, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214140517300014

[5] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: <italic>common practices and emerging technologies</italic>," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[6] 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, p. 628–647, Jul 2014. [Online]. Available: http://dx.doi.org/10.1002/oca.2123

[7] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," in *NeurIPS 2019 Competition and Demonstration Track*. PMLR, 2020, pp. 77–89.

[8] A. Liniger, "Pushing the limits of friction: A story of model mismatch," 2021, iCRA21 Autonomous Racing. [Online]. Available: https://www.youtube.com/watch?v=_rTawyZghEg&t=136s

[9] L. P. Fröhlich, C. Küttel, E. Arcari, L. Hewing, M. N. Zeilinger, and A. Carron, "Model learning and contextual controller tuning for autonomous racing," 2021.

[10] A. Jain, M. O'Kelly, P. Chaudhari, and M. Morari, "BayesRace: Learning to race autonomously using prior experience," *arXiv:2005.04755 [cs, eess]*, Nov. 2020, arXiv: 2005.04755. [Online]. Available: http://arxiv.org/abs/2005.04755

[11] E. Chisari, A. Liniger, A. Rupenyan, L. V. Gool, and J. Lygeros, "Learning from simulation, racing in reality," 2021.

[12] A. Brunnbauer, L. Berducci, A. Brandstätter, M. Lechner, R. Hasani, D. Rus, and R. Grosu, "Model-based versus Model-free Deep Reinforcement Learning for Autonomous Racing Cars," *arXiv:2103.04909 [cs]*, Mar. 2021, arXiv: 2103.04909. [Online]. Available: http://arxiv.org/abs/2103.04909

[13] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Durr, "Super-human performance in gran turismo sport using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, p. 4257–4264, Jul 2021. [Online]. Available: http://dx.doi.org/10.1109/LRA.2021.3064284

[14] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, L. Gilpin, P. Khandelwal, V. Kompella, H. Lin, P. MacAlpine, D. Oller, T. Seno, C. Sherstan, M. D. Thomure, H. Aghabozorgi, L. Barrett, R. Douglas, D. Whitehead, P. Dürr, P. Stone, M. Spranger, and H. Kitano, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, Feb. 2022. [Online]. Available: https://doi.org/10.1038/s41586-021-04357-7

[15] 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." [Online]. Available: https://arxiv.org/abs/2202.07008

[16] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenét frame," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 987–993.

[17] Y. Song, H. Lin, E. Kaufmann, P. Duerr, and D. Scaramuzza, "Autonomous Overtaking in Gran Turismo Sport Using Curriculum Reinforcement Learning," *arXiv:2103.14666 [cs]*, May 2021, arXiv: 2103.14666. [Online]. Available: http://arxiv.org/abs/2103.14666

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[19] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016. [Online]. Available: https://arxiv.org/abs/1606.01540

[20] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. Los Angeles, CA, USA: IEEE, Jun. 2017, pp. 719–726. [Online]. Available: http://ieeexplore.ieee.org/document/7995802/

[21] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 812–818.

[22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *CoRR*, vol. abs/1801.01290, 2018. [Online]. Available: http://arxiv.org/abs/1801.01290

[23] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html