Unifying F1TENTH Autonomous Racing: A Survey, Methods and Benchmark Results

Benjamin David Evans[1], Raphael Trumpp[2], Marco Caccamo[2], Hendrik Willem Jordaan[1], and Herman Arnold Engelbrecht[1].

***Abstract*—The F1TENTH autonomous racing platform, consisting of 1:10 scale RC cars, has evolved into a learning research platform. The many publications and real-world competition approaches span many domains, from classical path planning to novel learning-based algorithms. Consequently, the field is wide and disjointed, hindering direct comparison of methods and making it difficult to assess the state-of-the-art. Therefore, we aim to unify the field by surveying current approaches, describing common methods and providing benchmark results to facilitate clear comparison and establish a baseline for future work. We describe particle filter localisation, trajectory optimisation and control, model predictive contouring control (MPCC), follow-the-gap and end-to-end reinforcement learning. We provide an open-source evaluation of benchmark methods and investigate overlooked factors of control frequency and localisation accuracy for classical methods and reward signal and training map for learning methods. The evaluation shows that the trajectory generation and control method achieves the fastest lap times, followed by the MPCC planner. Finally, our work identifies and outlines the relevant research aspects to help motivate future work in the F1TENTH domain.**

***Index Terms*—F1TENTH, Autonomous Racing, Review, Deep reinforcement learning, Trajectory optimisation**

I. Introduction

The growing field of F1TENTH autonomous racing offers an exciting testbed for cutting-edge robotics and autonomous vehicle research [1]. The domain encompasses diverse aspects of autonomous systems, including perception, planning, control and learning [2]. However, the breath of F1TENTH research has left the field divided and fragmented, with researchers often focusing on specific subproblems in isolation. A particular divide exists between classical and machine learning approaches. This fragmentation hinders comparison between methods, making it difficult to assess algorithmic advancement. This paper aims to unify F1TENTH research by providing a comprehensive overview of the field and introducing a consistent perspective to unite researchers from different disciplines.

Autonomous racing presents a uniquely demanding testbed for the advancement of autonomous robotics methods. The racing problem is to use raw sensor data (LiDAR scans) to select control actions (speed and steering commands). The nature of racing provides a difficult challenge due to non-linear tyre dynamics, unstructured sensor data and real-time computing requirements [3]. A key challenge is the fundamental conflict between maximising performance and ensuring safety [4]. Higher speeds increase operational risk while prioritizing safety sacrifices competitive advantage. These difficulties promote advancement in high-performance, safe, and efficient autonomous vehicle systems.

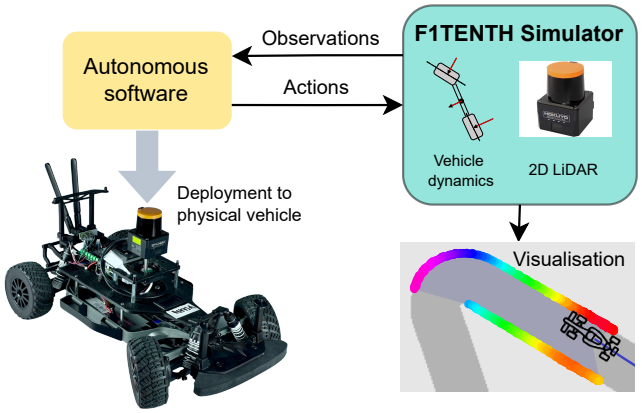

Fig. 1. The F1TENTH platform provides a development platform for autonomous algorithms with a good simulator and simplified deployment.

The F1TENTH platform is ideal for research because it enables rapid prototyping of algorithms, provides a comparable, competitive environment and simplifies software-to-hardware deployment [1]. Fig. 1 shows how racing software can be developed and tested in simulation before being deployed to a standardised physical vehicle [5]. The smaller size of F1TENTH cars makes them cost-effective and safe for developing new high-performance algorithms where crashes are inevitable. Finally, it provides a fun topic for students worldwide [6], and competitions encourage novel research.

We unify the field of F1TENTH racing by creating a common solution taxonomy through the following contributions:

- Survey advances in classical and learning-based approaches for F1TENTH racing in Section II.
- Describe common classical and reinforcement learning baseline methods in Section III.
- Provide benchmark results (with open-source software[1]) to accelerate future work in Section IV.
- Motivate promising research directions for future work on F1TENTH cars in Section V.

[1] Faculty of Electrical Engineering, Stellenbosch University, Stellenbosch, 7600, South Africa, {bdevans, wjordaan, hebrect}@sun.ac.za.
[2] TUM School of Engineering and Design, Technical University of Munich, Germany, {raphael.trumpp, mcaccamo}@tum.de
Marco Caccamo is supported by an Alexander von Humboldt Professorship endowed by the German Federal Ministry of Education and Research.
Manuscript received February 23, 2024

[1]Code available at: https://github.com/BDEvan5/f1tenth_benchmarks

II. Literature Survey

A. Classic Approaches

Classical approaches to autonomous racing use estimation, optimisation and control systems to calculate control commands. Typically, estimation is used to localise the vehicle on the map, optimisation is used for planning, and tracking algorithms are used to follow the trajectory. Fig. 2 shows how classical pipelines use localisation with either offline planning and online control or online planning and control. We study works related to each of these functions.

Perception: Perception is the task of building a map and localising the vehicle on the map. While no methods of map building, i.e., simultaneous localization and mapping (SLAM), have been built explicitly for F1TENTH racing, the ROS SLAM toolbox (using graph SLAM) is commonly used [7], [8]. Walsh et al. [9] implement a computationally efficient method for particle filter localization. Their approach, and open-source repo, is the standard localization method used in many other approaches [10], [11]. An extension of their approach has improved the original implementation by using a higher-order motion model [12]. A different approach using invertible neural networks to learn a localisation policy boasted improved computational performance [13]. However, the particle filter remains the most commonly used method due to its simplicity and robustness.

Offline trajectory planning: Offline trajectory planning uses an optimisation algorithm and map of the track to generate an optimal set of waypoints. The baseline planning method for most works uses a minimum curvature or minimum time trajectory planning formulation from Heilmeier et al. [14], which formulates the trajectory planning problem as a quadratic programming problem. Other approaches have turned to genetic programming algorithms to jointly optimise the trajectory and the controller [10], [15], but these can lead to simulator exploitation and unrealistic behaviour [16]. Heilmeier et al.'s method is the most popular because it is efficient to compute, robust to different tracks, open-source accessible and provides high-quality results [17], [18].

Online trajectory planning: At each planning step, online trajectory planning approaches generate a set of control inputs and a corresponding trajectory for a receding horizon [19]. The most popular approach is to use a model predictive controller (MPC), which optimises a trajectory to track a centre line [20] or an offline-generated trajectory [21]. MPC has been used for online system identification [20] and has the advantage of being able to update the vehicle model as more data is collected [21], [22]. Since MPC algorithms replan at each timestep, they can more easily integrate obstacle avoidance strategies [23], [24]. Due to the complexity of implementing optimisation routines, there is little comparison between MPC formulations (such as between different dynamics models) or other methods.

Control: The pure pursuit algorithm is a geometric approach to path following based on the vehicle dynamics [25], and has been widely implemented in many works [10], [26], [27]. An improvement on the original algorithm of using a varying length lookahead distance based on vehicle speed was proposed by [11]. The only paper that makes a significant contribution to the control of scale vehicles is the model-and-acceleration-based pure pursuit controller [18]. Their approach uses system identification of the non-linear dynamics to look up the control input that will result in the desired angular speed. This approach requires building a vehicle-specific look-up table but is currently state-of-the-art for physical vehicle control.

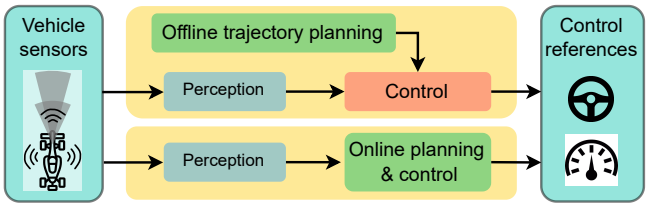

Fig. 2. Classical approaches use either offline planning with a separate control module, or online planning and control.

Mapless Methods: A commonly used mapless approach for F1TENTH racing is the Follow-the-Gap (FTG) controller. Proposed by Sezer et at. [29], this is a reactive method based on identifying gaps in the car's LiDAR signal. After finding the gaps, the car's steering angle is directly defined as the angle in direction or the widest or furthest. An extension to FTG is the disparity extender [30]. In this approach, before detecting the gaps in the LiDAR signal, all disparities in the LiDAR signal are identified. The distance readings of the LiDAR signal adjacent in some distance to the disparity are set to the minimum distance in each interval. Typically, the interval is chosen as the vehicle width plus a safety margin, thus avoiding trajectories that cut obstacles closely.

B. Learning Approaches

Deep learning approaches use neural networks to replace a part or all of the racing pipeline. We study learning approaches in the categories of architecture, algorithm and reward signal. The deep learning architecture determines the role of the agent in the racing pipeline. The algorithm denotes the process of training the neural network from random initialisation to a policy that controls the vehicle. The reward signal is the method of communicating the desired behaviour to the agent.

Learning architectures: The learning architecture describes the role of the agent and is impacts the input (state vector) given to the agent and the role/use of the actions generated. Fig. 3 shows the end-to-end, planning, and residual agents and safe learning architecture for autonomous racing. We consider approaches using each architecture.

End-to-end learning uses a neural network to replace the entire racing pipeline. In F1TENTH racing, common inputs for end-to-end agents are single or multiple downsampled 2D LiDAR scans and the vehicle speed [7], [35]. Many studies have addressed constant speed racing (driving) [32], [47], and implemented their approaches on physical vehicles at low, constant speeds [35], [36]. End-to-end approaches generalise well to unseen tracks [36] and transfer to physical vehicles [7]. A major challenge for end-to-end learning agents is reliable performance at high speed [37].

TABLE I
METHODS OF F1TENTH RACING IN THE LITERATURE, THE EVALUATION PLATFORM AND COMPARISONS USED

	Reference	Method/Novelty	Category	Simulation	Comparison	Real World
Classic	O'Kelly et at. (2020) [10], [16]	Genetic optimisation	Offline planing	F1TENTH Gym	Gradient-based trajectory optimisation [14]	✓
	Li et al. (2022) [23]	Multi-vehicle MPC	Online planning	Custom	✗	
	Jain et al. (2020) [22]	MPC with gaussian process	Control	F1TENTH Gym	MPC	✗
	Cataffo et al. (2022) [24]	MPC	Control	Gazebo [28]	✗	
	Wang et al. (2021) [20]	Data-driven MPC	Control	Gazebo	MPC, adaptive pure pursuit	✓
	Nagy et al. (2023) [21]	Adaptive MPC	Control	F1TENTH Gym	Pure pursuit	✓
	Sukhil and Behl (2021) [11]	Adaptive pure pursuit	Control	Gazebo [28]	Pure pursuit	✓
	Becker et al. (2023) [18]	Model-and-acceleration based pure pursuit	Control	✗	Pure pursuit	✓
	Sezer et at. (2012) [29]	Follow-the-gap	Mapless	Custom	APF, A*	✓
	Otterness (2019) [30]	Disparity extender	Mapless		✗	✓
Learning	Sun et al. (2023) [31]	New IL algorithm	End-to-end	F1TENTH Gym	IL, experts	✗
	Hamilton et al. (2022) [32]	DRL vs IL	End-to-end	Gazebo	✗	✓
	Cai et al. (2021) [33]	Vision-based IL	End-to-end	CARLA	IL, DRL	✓
	Sun et al. (2023) [34]	IL algorithm comparison	End-to-end	F1TENTH Gym	IL, DRL	✓
	Brunnbauer et al. (2022) [35]	Model-based DRL	End-to-end	PyBullet	Model-free DRL	✗
	Bosello et al. (2022) [36]	Generalisable DRL (DQN)	End-to-end	Rviz	DRL, Dreamer, FTG	✓
	Evans et al. (2023) [37]	Trajectory-aided learning	End-to-end	F1TENTH Gym	DRL	✗
	Evans et al. (2023) [7]	DRL	Comparison	F1TENTH Gym	DRL	✓
	Tătulea-Codrean et al. (2020) [38]	IL to replace MPC	Learned planning	Gazebo	MPC	✗
	Ghignone et al. (2022) [39]	Trajectory-conditioned learning	Learned planning	F1TENTH Gym	MPC, DRL	✗
	Dwivedi et al. (2022) [27]	Plan-assisted DRL	Learned planning	PyBullet	Dreamer, MPO	✗
	Trumpp et al. (2023) [40]	RPL for high-speed racing	RPL	F1TENTH Gym	Pure pursuit	✗
	Evans et al. (2021) [26], [41]	Obstacle avoidance	RPL	Custom	FTG, end-to-end, optimal plan	✗
	Zhang et al. (2022) [42]	APF-based RPL	RPL	PyBullet	FTG, MPC, DRL, Dreamer, experts	✗
	Trumpp et al. (2024) [43]	Mapless RPL racing	RPL	F1TENTH Gym	APF, disparity	✗
	Ivanov et al. (2020) [44]	Formal verification	Safe learning	Custom	DRL	✓
	Musau et al. (2022) [45]	Reachability Theory safety	Safe learning	Gazebo	IL, DRL	✓
	Evans et al. (2023) [46], [47]	Viability Theory safety	Safe learning	F1TENTH Gym	DRL, pure pursuit	✗

Planning agents learn a policy that incorporates the vehicle's location and upcoming track (usually in the form of centre points) into the state vector [27], [39]. Including the track points in the state has been shown to outperform end-to-end agents in training stability, sample efficient and racing performance [7]. Another motivation is to replace an online trajectory planner (MPC) with a neural network to reduce the computational burden [38], [48]. The limitation of planning agents is they require localisation, restricting them to mapped environments and putting them in direct competition with optimisation approaches.

Residual policy learning learns an additional policy that is added to a classical base policy [40]. Residual policies for racing have been learned on top of pure pursuit controllers [40] and artificial potential fields [42], resulting in faster convergence and improved stability. Residual policy learning has been extended to obstacle avoidance [26] and multi-agent overtaking manoeuvres [43]. Learning residuals promise to harness the flexibility of neural networks to improve upon classical components.

Safe learning is a method for training neural network controllers while simultaneously ensuring vehicle safety [46]. Safe learning approaches use a supervisor (action mask) to ensure that the controls executed on the vehicle are safe. Actions masks have used formal methods [44], reachability theory [45], time-to-collision (TTC) [36] and Viability Theory [46] to select which actions are safe. Safe learning enables safe deployment on physical hardware, resulting in many approaches having physical evaluations [36], [44], [45], [47]. A challenge in safe learning is building action masks that accurately discern safe and unsafe actions for high-performance systems.

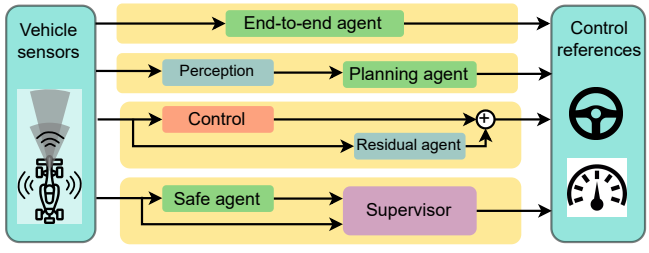

Fig. 3. End-to-end, planning, residual, and safe agent architectures for autonomous racing.

Training algorithms: While the majority of learning approaches are trained with reinforcement learning (RL) [7], [36], [39], imitation learning (IL) has also been used [31], [32], [34], [49]. Reinforcement learning trains agents to control a vehicle in an environment from the experience of receiving a state, selecting an action and receiving a reward. In contrast, imitation learning trains the network to mimic the behaviour of an expert, as recorded in a dataset. A study comparing the two found that RL is more robust to unseen states [32], a result backed up by the poor completion rates shown by IL algorithms in [34]. While attempts to combine IL and RL have been made, they are rare [33].

For training agents, the most common DRL algorithms have been TD3 [37], [44], SAC [32], [39], and PPO [40], [42]. Other algorithms that have been used are the deep-Q-networks (DQN) [36], and Dreamer, a model-based algorithm [35]. TD3 has shown to outperform SAC and DDPG for continuous control in autonomous racing [7]. It remains to be studied how off-policy algorithms (TD3) compare to on-policy algorithms (PPO) for autonomous racing.

Rewards signals: There has been much variation in the reward signals used to train DRL agents for autonomous racing. The reward signal plays a crucial role in the training process, but is often simply listed as a minor implementation detail. One of the most common is the progress reward signal, which rewards vehicles according to their progress along the track [35]. However, due to its lack of ability to communicate speed information, rewards that include the vehicle speed, and punish lateral deviation from the centre line have been favoured [36], [41]. A trajectory-aided learning (TAL) reward has been proposed that uses an optimal trajectory to train the agent [37]. Reward signal design for autonomous racing is difficult because the agent must be encouraged to race at high-speeds, while not being so aggressive that the vehicle crashes.

C. Literature Summary

Table II-A presents a tabular study of approaches to F1TENTH racing. While gradient-based offline trajectory optimisation methods, such as from [14] are widely used [18], no studies have designed such methods specifically for F1TENTH racing. Many papers have studied control methods for autonomous racing, focusing on using MPC for online control. A persistent challenge in high-speed racing is the identification and control of the non-linear dynamics.

There are many approaches in all learning categories for F1TENTH racing, with end-to-end being the largest. While learned planning has shown to be more robust, end-to-end methods have been tested on physical platforms a lot more. Few papers compare classical and learning-based methods, demonstrating the field's fragmentation. It is proposed that this is due to differing motivations for different methods; classical methods aim for high performance while learning methods aim for generality. One place of unity is that many approaches use the F1TENTH Gym simulator. This study demonstrates the need for better comparisons between methods.

Classical racing approaches using estimation, optimisation and control have proven highly effective in many contexts, offering both high performance and safety guarantees. However, their reliance on accurate dynamics models and apriori track knowledge poses a restriction on their applicability in unmodelled and unmapped contexts. In contrast, the follow-the-gap method requires neither a map nor a dynamics model, but often compromises performance. End-to-end reinforcement learning agents excel in unmapped settings, allowing neural networks to learn generalisable racing policies. Yet, concerns about the robustness and the complexity of their training pipelines persist. Recent efforts to combine the adaptability of neural networks with the reliability of classical methods, such as those seen in planning, residual, and safe learning agents, show promise but have yet to achieve widespread adoption.

III. Racing Methods

We describe some of the most common racing algorithms from each category to provide an overview of different solution approaches. We describe the particle filter algorithm used for localisation, a trajectory planning and control strategy, the model predictive contouring algorithm, the follow-the-gap method and end-to-end deep reinforcement learning.

A. Particle Filter Algorithm

The particle filter (PF) algorithm uses the control input and the 2D LiDAR scan to estimate the vehicle's pose on the map [9]. The algorithm is a recursive Bayesian state estimator designed for non-parametric distributions by representing the belief of the vehicle's pose by a set of particles.

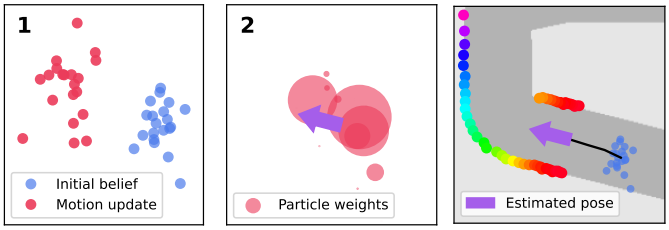

Fig. 4. Motion and measurement updates for the particle filter.

Fig. 4 shows the particle filter process for a vehicle in the environment shown on the right. A set of particles (blue) represents the belief of the vehicle's location. The motion update uses the control input and a vehicle model are used to update the particles to form the posterior belief (red). After each measurement from the LiDAR scan, weights (orange) are calculated for each particle based on the probability of the measurement being correct. The current pose (purple arrow) is estimated by multiplying the particle locations with the weights.

B. Trajectory Optimisation and Tracking

We consider a two-stage planner that uses the trajectory optimisation method presented by [14] and the pure pursuit path tracking algorithm from [25]. The two-stage nature means a trajectory can be calculated offline and tracked during the race.

Trajectory optimisation: The input into the optimisation is a list of centre line points and track widths from the map. Fig.

5 shows a track segment indicating the centre-line points used to calculate the heading angle ψ and segment length L. The minimum curvature path generation minimises the change in heading angle over the segment length.

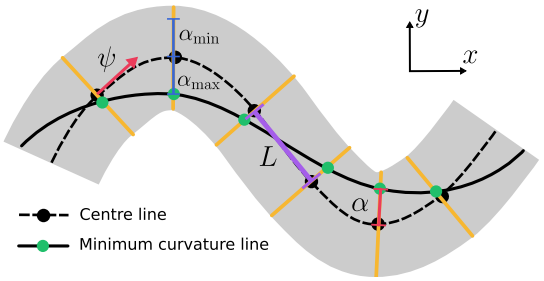

Fig. 5. Centre line, minimum curvature line, heading angle ψ, segment length L, and optimisation variable α and limits $[\alpha_{\min}, \alpha_{\max}]$ for a track segment.

The optimisation variable is α, which is the (positive or negative) distance from the track centre measured along the normal vector (perpendicular to the heading angle). The minimum and maximum values for α are defined by the track widths. The optimisation can be written as,

$$\text{minimise}_{[\alpha_1 \dots \alpha_N]} \quad \sum_{i=1}^{N} \left(\frac{d\psi}{dL} \right)^2 \tag{1}$$

$$\text{subject to} \quad \alpha_i \in [\alpha_{i,\min}, \alpha_{i,\max}] \tag{2}$$

The minimum time speed profile is calculated as the maximum speed that keeps the vehicle within the handling limits. For each point on the path i, the largest speed for the next point $v_{\text{i+1}}$ is calculated as,

$$v_{i+1} = \sqrt{v_i^2 \pm 2 l_i \cdot \mu a_{\max}(1 - v_i \kappa_i)}. \tag{3}$$

The inputs into the calculation are the maximum acceleration (based on the motors) current speed v_i, segment length l_i, maximum vehicle acceleration $a_{\max}$, path radius r_i and friction coefficient μ. The $\pm$ indicates that the process is repeated in both forward and reverse path directions.

Pure pursuit control: The pure pursuit path following algorithm tracks a reference path using a lookahead distance to select a point to steer towards. Given a lookahead point that is a lookahead distance l_{d} away from the vehicle, and at a relative heading angle of ψ, the steering angle δ is calculated as,

$$\delta = \arctan\left(\frac{L \sin(\psi)}{l_{\text{d}}} \right), \tag{4}$$

where L is the length of the vehicle's wheelbase. The lookahead distance l_{d} has constant and speed-dependant parts.

C. Model Predictive Contouring Control

Model predictive control is a receding horizon strategy that calculates a set of control inputs that optimise an objective function subject to constraints [4]. Model predictive contouring control (MPCC) maximises the progress along the centre line subject to the path being on the track and dynamically feasible.

The optimisation variables consist of the state and control inputs at each discrete timestep. The state is represented by position x, y and heading θ, and the control actions are speed v and steering δ. The state of centre line progress s and corresponding control of centre line speed $\dot{s}$ are added, resulting in a 4D state $[x, y, \theta, s]$ and 3D control $[\delta, v, \dot{s}]$. The initial state is set to the vehicle's position, and subsequent states are constrained to be the result of the control actions according to the kinematic bicycle model.

Objective: The MPCC objective maximises progress along the path using four reward terms: contouring error, lag error, progress, and control regularisation. Fig. 6 shows how the contouring and lag errors are measured.

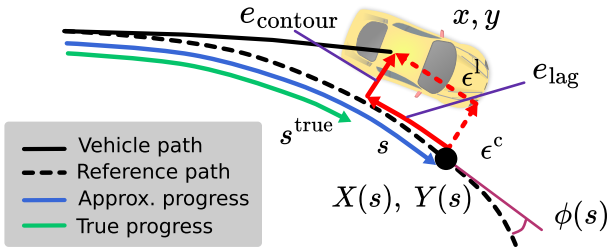

Fig. 6. Lag and contouring errors represent vehicle progress along the path.

The x and y coordinates and heading angle of the track are represented as functions of progress along the centre line path $X(s)$, $Y(s)$, $\phi(s)$. Since the true progress along the reference path s^{true} cannot be found, an approximate progress s is used. Fig. 6 shows how the lag error quantifies the difference between the true and approximate progress. The contouring error quantifies the vehicle's perpendicular distance from the trajectory. The linearised contouring and lag errors (ϵ^c and ϵ^l) can be calculated as,

$$\epsilon^{\text{c}} = \sin(\phi(s))(x - X(s)) - \cos(\phi(s))(y - Y(s)) \tag{5}$$

$$\epsilon^{\text{l}} = -\cos(\phi(s))(x - X(s)) - \sin(\phi(s))(y - Y(s)) \tag{6}$$

The progress speed along the centre line is rewarded to encourage the vehicle to maximise progress along the track. Finally, a control regularisation term penalises steering angles, encouraging smooth behaviour. Each term is weighted, and the weights are tuned to ensure satisfactory performance.

D. Follow-the-gap Method

The follow-the-gap (FTG) method, also known as the disparity extender, is a simple, mapless approach to autonomous racing [29], [30]. The algorithm identifies the nearest boundary/obstacle in the LiDAR scan and excludes a bubble of the beams around it. Then, the largest visible gap in the LiDAR scan is identified, and the steering angle is calculated to drive towards the middle of the gap. The speed is selected by using a higher speed (5 m/s) for small steering angles and a low speed (3m/s) for high steering angles.

E. End-to-end Deep Reinforcement Learning

End-to-end learning agents for autonomous racing use a neural network to map a state vector s directly to a control

action a [7]. Mathematically, the mapping is written as a policy π that maps a state to an action as $a = \pi(s)$. The policy is a deep neural network of multiple fully connected layers. The state vector is comprised of the previous and current LiDAR scans and the vehicle speed. The control actions are the speed and steering angle used to control the vehicle.

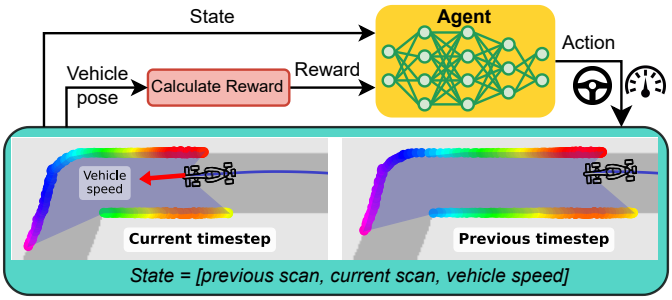

Fig. 7. Traing setup for training a reinforcement learning agent to race in a simulator.

Fig. 7 shows the training setup for an end-to-end reinforcement learning agent. During training, the agent experiences states, selects actions and receives rewards. The reward, which indicates how good or bad an action is, is calculated based on the vehicle's pose and/or the agent's action. The reinforcement learning objective is to select actions that maximise the total reward [50]. The training is split into episodes where the vehicle is randomly spawned on the track and must drive until it either crashes or completes a lap.

We use the TD3 algorithm [51], a continuous control, off-policy, actor-critic reinforcement learning algorithm. The actor network is the policy that selects actions, and the critic network learns a Q-value function for each state-action pair. The critic is trained towards targets calculated by the Bellman update equation. The actor is updated using the policy gradient that maximises Q-values.

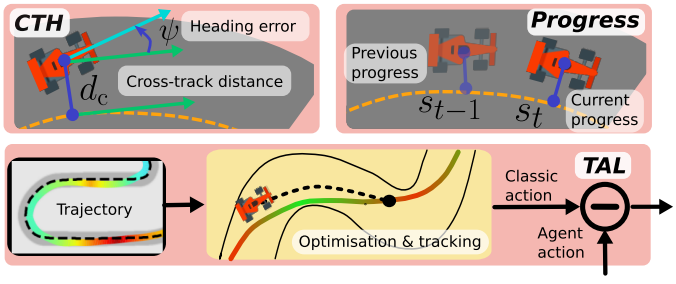

Fig. 8. Illustrations of the cross-track and heading, track progress and trajectory-aided learning reward signals.

To train the agent to complete laps without crashing, the agent is rewarded with 1 for completing a lap, punished with -1 if the vehicle crashes, and given a shaped reward to encourage fast racing behaviour. Fig. 8 illustrates the three shaped reward signals of cross-track and heading error (CTH), track progress and trajectory-aided learning (TAL).

The *cross-track and heading error (CTH)* reward punishes the cross-track distance d_c and promotes speed in the direction of the track. The normalised vehicle speed v_t, heading error ψ and cross-track error d_c are used to calculate the reward as,

$$r_{\text{CTH}} = v_t \cos\psi - d_c, \tag{7}$$

The *track progress* reward promotes advancement along the track centre line by rewarding the progress made during each timestep. The current progress s_t and previous progress s_{t-1} are used to calculate the reward as,

$$r_{\text{progress}} = s_t - s_{t-1}, \tag{8}$$

The *trajectory-aided learning (TAL)* reward trains the agent to mimic the actions selected by the optimisation and tracking planner described in §III-B [37]. The agent's action $\mathbf{u}_{\text{agent}}$ and classic action (trajectory planning and control) $\mathbf{u}_{\text{classic}}$ is used to calculate the TAL reward as,

$$r_{\text{TAL}} = 1 - |\mathbf{u}_{\text{agent}} - \mathbf{u}_{\text{classic}}| \tag{9}$$

IV. Benchmark Evaluation

We benchmark the results described in Section III by,

1) Investigating the impact of control frequency and localisation error on racing performance
2) Comparing the CTH, progress and TAL reward signals, and training maps on agent performance
3) Providing benchmark results for future studies

For each test, we present the key quantitative result (i.e. lap time) and then qualitative data (i.e. speed profile plots) to explain the reasons for the result.

A. Methodology

We use the open-source[2] F1TENTH autonomous vehicle simulator, as presented by O'Kelly et al. [5]. The vehicle is represented by the single-track bicycle model (described in Appendix A), which describes the state in terms of position, speed, steering angle, yaw, yaw rate, and slip angle [52]. The 2D LiDAR scanner is simulated with a ray-casting algorithm that measures the distance to the track boundary for each beam. The simulator and all the tests are written in Python and run on an Ubuntu computer. The simulator dynamics are updated at a rate of 100 Hz in all tests, and unless otherwise stated, the planning frequency is 25 Hz.

We use a set of four F1 race tracks that have been rescaled ($\approx$ 1:20) for F1TENTH racing and had the widths adjusted to a constant value. These maps have been widely used in the literature [7], [35], [36], since they provide a difficult challenge for F1TENTH vehicles and allow for results to be easily compared. Fig. 9 shows the shape of the circuits in Austria (AUT), Spain (ESP), Great Britain (GBR) and Monaco (MCO). Appendix B provides the track length, percentage of straight sections and number of corners for each track.

For all tests, 10 laps are run with starting points randomly selected along the centre line. All the results are seeded for reproducibility. The following metrics are used,

- Lap time: the mean of the lap times for successfully completed laps.

[2]https://github.com/f1tenth/f1tenth_gym

Fig. 9. Maps of the AUT, ESP, GBR, and MCO race tracks (left to right).

- Completion rate: the number of successfully completed laps divided by the total number of test laps
- Progress: the progress made by the vehicle before crashing (used for learning tests)

The speed and slip angle profiles and the path taken are analysed to better understand the results by exposing the vehicle behaviour.

B. Control Frequency Evaluation

We investigate the impact of control frequency and localisation error on racing performance. We use the optimisation and tracking planner with the true location and the estimated pose from the particle filter. We use control frequencies ranging from 7.1 to 50 Hz and friction coefficients (for the raceline generation) of 0.7, 0.8, 0.9 and 1. For these tests, we use 300 particles in the particle filter estimation to expose the effect of inaccurate localisation.

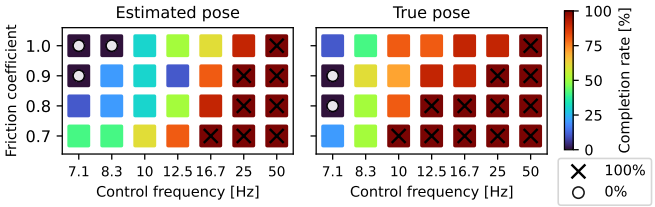

Fig. 10. Friction and control frequency plotted against completion rate for the optimisation and tracking planner using the estimated and the true poses.

Fig. 10 shows the completion rates, with both planners achieving higher success rates at high control frequencies and lower success rates at lower frequencies. The planner with the true pose achieves a 100% completion rate at a control frequency of 10 Hz, while the estimated pose only at 16.7 Hz. For the estimated pose results using a control frequency of 16.7 Hz, friction coefficients of 0.7 and 1 result in 100% and 60% completion rates, respectively. Using a higher friction coefficient results in the planner requiring a higher control frequency to complete laps.

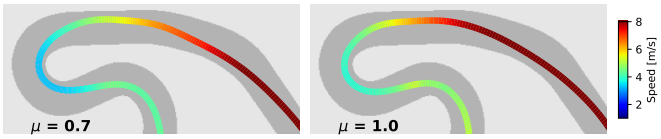

Fig. 11. Trajectories on the GBR using friction coefficients of 0.7 and 1.0

Fig. 11 shows trajectory segments on the GBR map using friction coefficients of 0.7 and 1.0 and a control frequency of 50 Hz. The right trajectory has a longer section of dark red compared to the left image, indicating a faster speed profile and thus requiring a higher control frequency.

Conclusion 1: Evaluations should study control frequency and error introduced by localisation because they significantly impact the results.

C. Training Configuration Evaluation

The learning evaluation compares the progress, CTH and TAL reward signals and investigates the impact of the training map on DRL agent performance. Three agents are trained for each reward signal on each map for 50,000 steps.

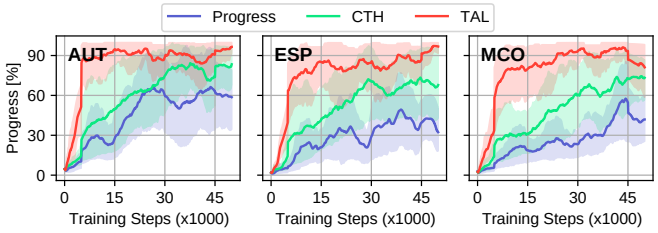

Fig. 12. Average progress during training on the AUT, GBR and MCO tracks.

Fig. 12 shows the mean lap progress during training, with the saded regions indicating the minimum and maximum. The TAL reward produces the fastest training, and the agents reach around 90% average progress in around 15k steps. The CTH reward produces slower training, and the agents converge to around 70% average progress. The progress reward results in the agents converging to low average progresses, around 50%.

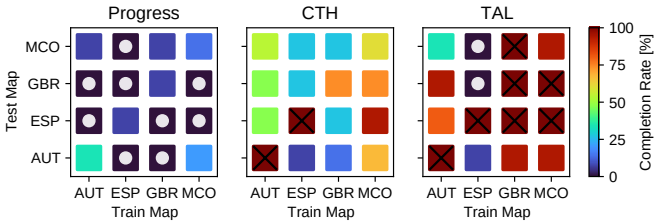

Fig. 13. Completion rate for DRL agents tested on each map.

Fig. 13 shows the completion rate for each agent for 5 test laps on each map. The progress agent achieves low completion rates on all the maps, with many combinations achieving a completion rate of zero. The CTH agents achieve higher completion rates, with some agents achieving 100%. The TAL agents achieve the highest completion rates with 7 of the tests achieving a 100% completion rate. The GBR map produces the best generalisation performance, with the TAL agents achieving 80-100% completion on all other maps.

Fig. 14 shows trajectory segments for agents trained and tested on MCO with the slip angle visualised as the red intensity. The progress and CTH agents have many times where the vehicle has a large slip angle, while the TAL agent maintains a low slip angle profile. This agrees with previous work [37], which proposed the reason for the TAL reward outperforming others is that it selects a speed profile similar to the optimal trajectory, resulting a lower slip angle.

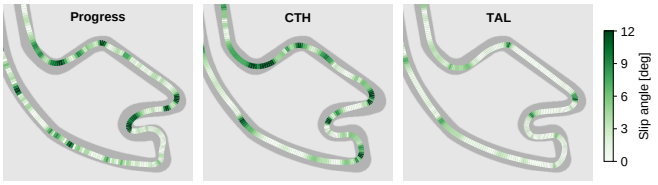

Fig. 14. Trajectory segments from MCO showing the absolute slip angle.

Conclusion 2: The TAL reward has the highest completion rate and produces the lowest slip angles.

D. Benchmark Results

We present benchmark results by providing lap times and key insights to explain the lap times. The optimisation and control, MPCC, follow-the-gap and end-to-end agents (trained on GBR with the TAL reward) are tested by running five test laps on each map and recording the average lap times.

Planner	**Map**			
	AUT	ESP	GBR	MCO
Planning + control	**16.88**	**36.17**	**31.54**	**28.84**
MPCC	18.36	42.29	39.31	34.92
Follow-the-gap	19.12	45.74	39.33	35.03
End-to-end	19.94	44.57	39.80	36.58

TABLE II
MEAN LAP TIMES [S] FOR THE OPTIMISATION AND TRACKING, MPCC, FOLLOW-THE-GAP AND END-TO-END METHODS.

Table IV-D presents the mean lap times for the planners on the four test maps. The optimisation and tracking planner achieved the fastest lap time on all the maps. The MPCC comes close second, being 2-6 s slower on each map. The follow-the-gap and end-to-end methods achieve similar lap times, significantly slower than the other two approaches.

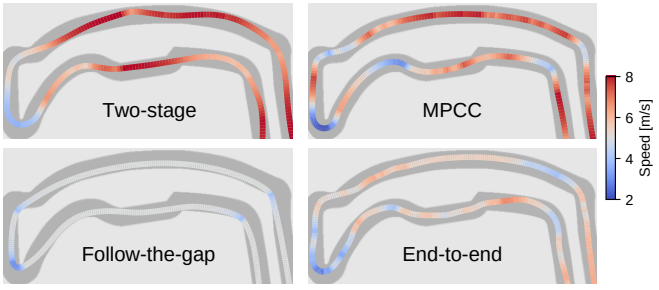

Fig. 15. Trajectories with colour indicating speed on the MCO track

To explain the lap time results, Fig. 15 shows trajectory segments for each planner on a section of the MCO map. The optimisation and tracking planner selects a smooth profile that slows down in the corners and speeds up in the straighter sections. The MPCC planner selects a more jagged speed profile by slowing down more in the corners. The follow-the-gap method often drives in a straight line and always cuts to the inside of the corner while slowing down. The end-to-end planner sometimes speeds up, but not to the full speed, and selects a varying speed profile that appears to change for no reason.

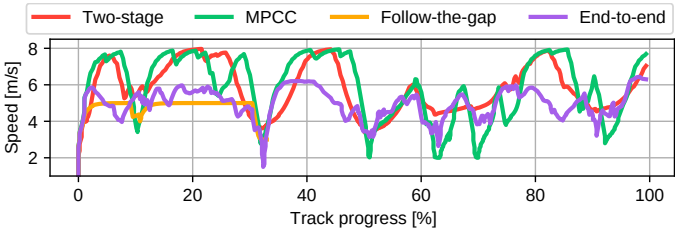

Fig. 16. Comparison of the speed profiles for the AUT map.

The graph in Fig. 16 shows the speed profiles for each planner on the AUT map. The poor performance on the follow-the-gap and end-to-end planners is clearly explained by them selecting low speeds for most of the lap. As seen in the trajectories in Fig. 15, the MPCC selects a more jagged speed profile, slowing down excessively. It is proposed that the slower speeds are due to the friction constraint on the MPCC algorithm, compared to the offline trajectory optimisation, which optimises a profile for the entire track.

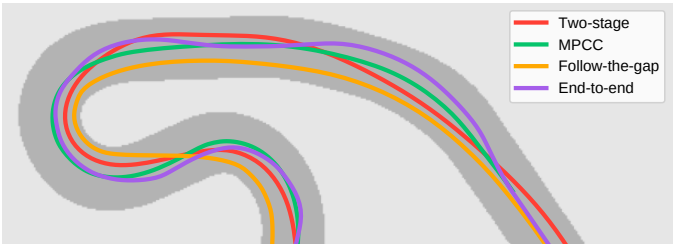

Fig. 17. Paths from a portion of the GBR track.

Fig. 17 shows the paths the planners took on a portion of the GBR map. The optimisation and control planner smoothly enters the corners, tracking the minimum curvature line. The MPCC planner tracks the centre line, fulfilling its objective of maximising centre line progress. The follow-the-gap method takes the shortest path, which is often straight and turns sharply at the corners. The end-to-end agent shows a more wobbly path and often staying near the middle of the track.

Conclusion 3: Offline trajectory optimisation and control achieves the fastest lap times for F1TENTH autonomous racing. It is followed by MPCC, with the follow-the-gap method and end-to-end agents being significantly slower.

V. CONCLUSION

This paper addressed the fragmented field of F1TENTH autonomous racing. The literature study described how different methods have been used for the subproblems of classical and learning-based approaches. The study explained how the methods relate to one another, creating a taxonomy for future work to fit into. The methods of particle filter localisation, trajectory optimisation and tracking, model predictive contouring control, follow-the-gap and end-to-end deep reinforcement learning were described. The evaluation highlighted the importance of reporting on the overlooked factors of control frequency and

localisation error for classical approaches, as well as reward signal and training map for learning methods. The benchmark evaluation concluded that trajectory optimisation and tracking solutions currently provide the fastest lap times due to optimal speed profile selection.

A. Future Work

Multi-agent benchmarks: F1TENTH racing is moving towards the multi-agent setting. Benchmark methods and standardised evaluations should be constructed to promise progress in that direction.

Vision-based racing: Almost all racing methods use LiDAR as the main input modality. While LiDAR's are convenient since they provide geometric information about the surrounding environment, they are expensive and poorly suited to real-world applications. In contrast, cameras are cheaper and contain more expressive information. Therefore, using cameras for high-performance control should be investigated.

Mapless solutions: A majority of approaches, especially in the categories of optimisation-based planning and control, assume that a map of the environment is available. While this is often the case in racing, it is not the case in many real-world applications. Therefore, research should investigate solutions that do not require a map, and thus promise flexibility.

REFERENCES

[1] M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio *et al.*, "F1/10: An open-source autonomous cyber-physical platform," *arXiv preprint arXiv:1901.08567*, 2019.

[2] 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*, 2022.

[3] J. Betz, T. Betz, F. Fent, M. Geisslinger, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, M. Lienkamp *et al.*, "Tum autonomous motorsport: An autonomous racing software for the indy autonomous challenge," *arXiv preprint arXiv:2205.15979*, 2022.

[4] 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.

[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*, vol. 123, 2020.

[6] J. Betz, H. Zheng, Z. Zang, F. Sauerbeck, K. Walas, V. Dimitrov, M. Behl, R. Zheng, J. Biswas, V. Krovi *et al.*, "Teaching autonomous systems hands-on: Leveraging modular small-scale hardware in the robotics classroom," *arXiv preprint arXiv:2209.11181*, 2022.

[7] B. D. Evans, H. W. Jordaan, and H. A. Engelbrecht, "Comparing deep reinforcement learning architectures for autonomous racing," *Machine Learning with Applications*, vol. 14, p. 100496, 2023.

[8] N. Gupta, K. Wilson, and Z. Guo, "Optimizing real-time performances for timed-loop racing under f1tenth," *arXiv preprint arXiv:2212.04549*, 2022.

[9] C. H. Walsh and S. Karaman, "Cddt: Fast approximate 2d ray casting for accelerated localization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3677–3684.

[10] M. O'Kelly, H. Zheng, A. Jain, J. Auckley, K. Luong, and R. Mangharam, "Tunercar: A superoptimization toolchain for autonomous racing," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5356–5362.

[11] V. Sukhil and M. Behl, "Adaptive lookahead pure-pursuit for autonomous racing," *arXiv preprint arXiv:2111.08873*, 2021.

[12] T. Y. Lim, E. Ghignone, N. Baumann, and M. Magno, "Robustness evaluation of localization techniques for autonomous racing," 2024.

[13] Z. Zang, H. Zheng, J. Betz, and R. Mangharam, "Local_inn: Implicit map representation and localization with invertible neural networks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 742–11 748.

[14] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, and B. Lohmann, "Minimum curvature trajectory planning and control for an autonomous race car," *Vehicle System Dynamics*, vol. 58, no. 10, pp. 1497–1527, 10 2020. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00423114.2019.1631455

[15] J. Klapálek, A. Novák, M. Sojka, and Z. Hanzálek, "Car racing line optimization with genetic algorithm using approximate homeomorphism," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 601–607.

[16] H. Zheng, J. Betz, and R. Mangharam, "Gradient-free multi-domain optimization for autonomous systems," *arXiv preprint arXiv:2202.13525*, 2022.

[17] J. Klapálek, M. Sojka, and Z. Hanzálek, "Comparison of control approaches for autonomous race car model," in *Proceedings of the FISITA 2021 World Congress*. FISITA-International Federation of Automotive Engineering Societies, 2021.

[18] J. Becker, N. Imholz, L. Schwarzenbach, E. Ghignone, N. Baumann, and M. Magno, "Model-and acceleration-based pursuit controller for high-performance autonomous racing," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5276–5283.

[19] A. Liniger, "Path Planning and Control for Autonomous Racing," Ph.D. dissertation, ETH Zurich, 2018.

[20] R. Wang, Y. Han, and U. Vaidya, "Deep koopman data-driven control framework for autonomous racing," in *Proc. Int. Conf. Robot. Autom.(ICRA) Workshop Opportunities Challenges Auton. Racing*, 2021, pp. 1–6.

[21] T. Nagy, A. Amine, T. X. Nghiem, U. Rosolia, Z. Zang, and R. Mangharam, "Ensemble gaussian processes for adaptive autonomous driving on multi-friction surfaces," *arXiv preprint arXiv:2303.13694*, 2023.

[22] A. Jain, M. O'Kelly, P. Chaudhari, and M. Morari, "Bayesrace: Learning to race autonomously using prior experience," in *Conference on Robot Learning*. PMLR, 2021, pp. 1918–1929.

[23] N. Li, E. Goubault, L. Pautet, and S. Putot, "A real-time nmpc controller for autonomous vehicle racing," in *2022 6th International Conference on Automation, Control and Robots (ICACR)*. IEEE, 2022, pp. 148–155.

[24] V. Cataffo, G. Silano, L. Iannelli, V. Puig, and L. Glielmo, "A nonlinear model predictive control strategy for autonomous racing of scale vehicles," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2022, pp. 100–105.

[25] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.

[26] 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)*. IEEE, 2021, pp. 601–606.

[27] T. Dwivedi, T. Betz, F. Sauerbeck, P. Manivannan, and M. Lienkamp, "Continuous control of autonomous vehicles using plan-assisted deep reinforcement learning," in *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2022, pp. 244–250.

[28] V. S. Babu and M. Behl, "F1tenth.dev-An Open-source ROS based F1/10 Autonomous Racing Simulator," in *IEEE International Conference on Automation Science and Engineering*, vol. 2020-Augus. IEEE Computer Society, 8 2020, pp. 1614–1620.

[29] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm:"follow the gap method"," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1123–1134, 2012.

[30] N. Otterness, "The "Disparity Extender" Algorithm, and F1/Tenth." [Online]. Available: https://www.nathanotterness.com/2019/04/the-disparity-extender-algorithm-and.html

[31] X. Sun, S. Yang, and R. Mangharam, "Mega-dagger: Imitation learning with multiple imperfect experts," *arXiv preprint arXiv:2303.00638*, 2023.

[32] N. Hamilton, P. Musau, D. M. Lopez, and T. T. Johnson, "Zero-shot policy transfer in autonomous racing: Reinforcement learning vs imitation learning," in *2022 IEEE International Conference on Assured Autonomy (ICAA)*. IEEE, 2022, pp. 11–20.

[33] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, "Vision-based autonomous car racing using deep imitative reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7262–7269, 2021.

[34] X. Sun, M. Zhou, Z. Zhuang, S. Yang, J. Betz, and R. Mangharam, "A benchmark comparison of imitation learning-based control policies for

autonomous racing," in *2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2023, pp. 1–5.
[35] A. Brunnbauer, L. Berducci, A. Brandstatter, M. Lechner, R. Hasani, D. Rus, and R. Grosu, "Latent Imagination Facilitates Zero-Shot Transfer in Autonomous Racing," *2022 International Conference on Robotics and Automation (ICRA)*, pp. 7513–7520, 5 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9811650/
[36] M. Bosello, R. Tse, and G. Pau, "Train in austria, race in montecarlo: Generalized rl for cross-track f1 tenth lidar-based races," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2022, pp. 290–298.
[37] B. D. Evans, H. A. Engelbrecht, and H. W. Jordaan, "High-speed autonomous racing using trajectory-aided deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, pp. 5353–5359, 2023.
[38] A. Tătulea-Codrean, T. Mariani, and S. Engell, "Design and simulation of a machine-learning and model predictive control approach to autonomous race driving for the f1/10 platform," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6031–6036, 2020.
[39] E. Ghignone, N. Baumann, and M. Magno, "Tc-driver: A trajectory conditioned reinforcement learning approach to zero-shot autonomous racing," *Field Robotics*, vol. 3, no. 1, pp. 637–651, 2023.
[40] R. Trumpp, D. Hoornaert, and M. Caccamo, "Residual policy learning for vehicle control of autonomous racing cars," in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–6.
[41] B. Evans, H. A. Engelbrecht, and H. W. Jordaan, "Reward signal design for autonomous racing," in *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, 2021, pp. 455–460.
[42] R. Zhang, J. Hou, G. Chen, Z. Li, J. Chen, and A. Knoll, "Residual policy learning facilitates efficient model-free autonomous racing," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 625–11 632, 2022.
[43] R. Trumpp, E. Javanmardi, J. Nakazato, M. Tsukada, and M. Caccamo, "Racemop: Mapless online path planning with residual policy learning for multi-agent autonomous racing," *Under submission*, 2024.
[44] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Case study: verifying the safety of an autonomous racing car with a neural network controller," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 2020, pp. 1–7.
[45] P. Musau, N. Hamilton, D. M. Lopez, P. Robinette, and T. T. Johnson, "On using real-time reachability for the safety assurance of machine learning controllers," in *2022 IEEE International Conference on Assured Autonomy (ICAA)*. IEEE, 2022, pp. 1–10.
[46] B. D. Evans, H. W. Jordaan, and H. A. Engelbrecht, "Safe reinforcement learning for high-speed autonomous racing," *Cognitive Robotics*, vol. 3, pp. 107–126, 2023.
[47] B. D. Evans, J. Betz, H. Zheng, H. A. Engelbrecht, R. Mangharam, and H. W. Jordaan, "Bypassing the simulation-to-reality gap: Online reinforcement learning using a supervisor," in *2023 21st International Conference on Advanced Robotics (ICAR)*. IEEE, 2023, pp. 325–331.
[48] M. Luiza Costa Vianna, E. Goubault, and S. Putot, "Neural Network Based Model Predictive Control for an Autonomous Vehicle," *arXiv e-prints*, p. arXiv:2107.14573, Jul. 2021.
[49] J. Zhang and H.-W. Loidl, "F1tenth: An over-taking algorithm using machine learning," in *2023 28th International Conference on Automation and Computing (ICAC)*. IEEE, 2023, pp. 01–06.
[50] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[51] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.
[52] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 719–726.
[53] ——, "Commonroad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 719–726.

VI. Biography Section

Benjamin David Evans studied for a bachelor's in Mechatronic Engineering at the University of Stellenbosch, graduating in 2019. He started a masters, focused on using reinforcement learning for autonomous racing, which was upgraded to a PhD. After graduating in 2023, he continues as a postdoctoral researcher at Stellenbosch University. His interests include the intersection of classical control and machine learning for autonomous systems.

Raphael Trumpp graduated with a M.Sc. degree in mechanical engineering from the Technical University of Munich in 2021, where he is currently pursuing a Ph.D. in informatics. His research focuses on machine learning, especially combining deep reinforcement learning with classical control methods. He is interested in applying these to interactive multi-agent scenarios like autonomous racing and robotics.

Marco Caccamo earned his Ph.D. in computer engineering from Scuola Superiore Sant'Anna (Italy) in 2002. Shortly after graduation, he joined University of Illinois at Urbana-Champaign as assistant professor in Computer Science and was promoted to full professor in 2014. Since 2018, Prof. Caccamo has been appointed to the chair of Cyber-Physical Systems in Production Engineering at Technical University of Munich, Germany. In 2003, he was awarded an NSF CAREER Award. He is a recipient of the Alexander von Humboldt Professorship and he is IEEE Fellow.

Hendrick Willem Jordaan received his bachelor's in Electrical and Electronic Engineering with Computer Science and continued to receive his Ph.D degree in satellite control at Stellenbosch University. He currently acts as a senior lecturer at Stellenbosch University and is involved in several research projects regarding advanced control systems as applied to different autonomous vehicles. His interests include robust and adaptive control systems applied to practical vehicles.

Herman Engelbrecht received his Ph.D. degree in Electronic Engineering from Stellenbosch University (South Africa) in 2007. He is currently the Chair of the Department of Electrical and Electronic Engineering. His research interests include distributed systems (specifically infrastructure to support massive multi-user virtual environments) and machine learning (specifically deep reinforcement learning). Prof Engelbrecht is a Senior Member of the IEEE and a Member of the ACM.

Appendix A
Single-Track Model

The F1TENTH-gym's simulation is based on a single-track model, also known as the bicycle model. The implementation is derived from the CommonRoad framework [53]. This model simplifies the vehicle kinematics to a two-wheel system allowing for under- and oversteering by introducing a slip angle β and a friction coefficient μ. For a global position $[s_x, s_y]$ with yaw angle Ψ as global orientation, the state is defined as

$$\vec{x} = \begin{bmatrix} s_x & s_y & \delta & v & \Psi & \dot{\Psi} & \beta \end{bmatrix}^\top, \tag{10}$$

with velocity v and steering angle δ.

Given v_δ as the velocity of the steering angle and the lateral acceleration a_{long} as input variables u_1 and u_2, respectively, the state derivatives are given by

$$\begin{aligned}
\dot{x}_1 &= x_4 \cos(x_5 + x_7), \\
\dot{x}_2 &= x_4 \sin(x_5 + x_7), \\
\dot{x}_3 &= u_1, \\
\dot{x}_4 &= u_2, \\
\dot{x}_5 &= x_6, \\
\dot{x}_6 &= \frac{\mu m}{I_z(l_r + l_f)} \big[l_f C_{S,f}(g l_r - u_2 h_{cg}) x_3 \\
&\quad + (l_r C_{S,r}(g l_f + u_2 h_{cg}) - l_f C_{S,f}(g l_r - u_2 h_{cg})) x_7 \\
&\quad - (l_f^2 C_{S,f}(g l_r - u_2 h_{cg}) + l_r^2 C_{S,r}(g l_f + u_2 h_{cg})) \frac{x_6}{x_4} \big], \\
\dot{x}_7 &= \frac{\mu}{x_4(l_r + l_f)} \big[C_{S,f}(g l_r - u_2 h_{cg}) x_3 \\
&\quad - (C_{S,r}(g l_f + u_2 h_{cg}) + C_{S,f}(g l_r - u_2 h_{cg})) x_7 \\
&\quad + (C_{S,r}(g l_f + u_2 h_{cg}) l_r - C_{S,f}(g l_r - u_2 h_{cg}) l_f) \frac{x_6}{x_4} \big] \\
&\quad - x_6.
\end{aligned} \tag{11}$$

In contrast to other works, this model from Althoff et al. [53] accounts for load transfer between axles and does not assume constant velocity. Therefore, the model is suitable for simulating evasive manoeuvres closer to the vehicle's physical limits. Note that this model becomes singular for small velocities, so a kinematic model is used for $|v| < 0.1$.

APPENDIX B
MAP CHARACTERISTICS

We quantify the characteristics of these maps by providing the length, straight percentage, and number of corners in Table III. The straight percentage is the length of the track with curvature below 0.1 rad/m divided by the total track length. The number of corners is the number of sections with curvature above 0.6 rad/m. The table shows that the ESP is the longest track with a length of 236.93 m, AUT has the highest % of straight sections and MCO has 16 corners, more than double the other maps.

Map name	AUT	ESP	GBR	MCO
Track length [m]	94.90	236.93	201.84	178.71
Straight length [%]	64.92	58.98	59.45	60.60
Corner count	7	7	7	16

TABLE III
QUANTIFIABLE TRACK CHARACTERISTICS

APPENDIX C
PARTICLE FILTER TUNING

We ran a test lap using the pure pursuit algorithm following the centre line at a low constant speed of 2 m/s, and evaluated the accuracy using increasing numbers of particles. We present Fig. 18 to indicate the impact of the number of particles on the localisation accuracy and computation time. The graph shows that the error drops from 10 cm for 50 particles to around 4 cm for 1000 particles. Using 1000 particles has a computation time of 10 ms, which is below our planning period of 40 ms. Therefore, we use 1000 particles for our tests.

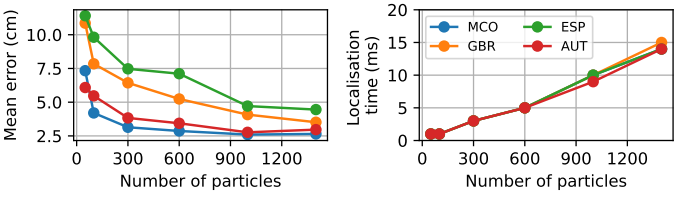

Fig. 18. The localisation error and computation time for varying numbers of particles.