

High-performance Racing on Unmapped Tracks using Local Maps

Benjamin David Evans¹, Hendrik Willem Jordaan¹ and Herman Arnold Engelbrecht¹

Abstract—Map-based methods for autonomous racing estimate the vehicle’s location, which is used to follow a high-level plan. While map-based optimisation methods demonstrate high-performance results, they are limited by requiring a map of the environment. In contrast, mapless methods can operate in unmapped contexts since they directly process raw sensor data (often LiDAR) to calculate commands. However, a major limitation in mapless methods is poor performance due to a lack of optimisation. In response, we propose the local map framework that uses easily extractable, low-level features to build local maps of the visible region that form the input to optimisation-based controllers. Our local map generation extracts the visible racetrack boundaries and calculates a centreline and track widths used for planning. We evaluate our method for simulated F1Tenth autonomous racing using a two-stage trajectory optimisation and tracking strategy and a model predictive controller. Our method achieves lap times that are 8.8% faster than the Follow-The-Gap method and 3.22% faster than end-to-end neural networks due to the optimisation resulting in a faster speed profile. The local map planner is 3.28% slower than global methods that have access to an entire map of the track that can be used for planning. Critically, our approach enables high-speed autonomous racing on unmapped tracks, achieving performance similar to global methods without requiring a track map.

I. INTRODUCTION

Methods for autonomous vehicles can be grouped into map-based or mapless methods. Classical map-based methods use a perception, planning, and control stack to estimate the vehicle’s pose, calculate an optimal trajectory and then track it [1]. Map-based methods are limited to contexts where maps that can be used for localisation exist. In contrast, mapless methods do not require a map since they calculate control commands directly from the incoming sensor measurements. Current mapless methods of reactive algorithms [2] and end-to-end neural networks [3] achieve poor performance and low completion rates. In response, this paper addresses the problem of high-performance racing on unmapped tracks.

Map-based methods are limited to contexts where an accurate map has been built, and localisation is available. Autonomous vehicles should be able to operate in contexts where maps are unavailable or where the environment has changed since the map was built. For example, self-driving cars must operate in GPS-denied environments where localisation on a map is not possible [4]. Mapless vehicle control is difficult due to the challenges in directly interpreting sensor data such as LiDAR scans or camera images. This

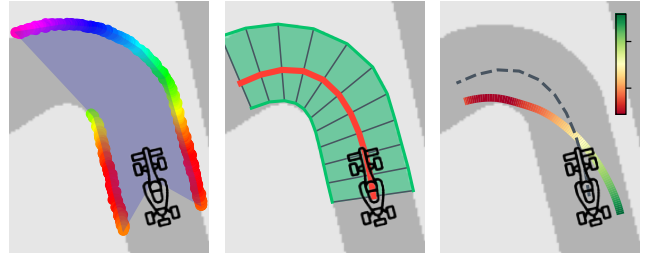


Fig. 1. Local map racing pipeline; (1) receive LiDAR scan, (2) extract LocalMap, (3) calculate optimal trajectory.

has resulted in most optimisation-based planning methods focusing on situations where prebuilt maps are available [5].

In this paper, we propose using easily extractable, low-level features to build local maps that can be used for optimisation-based planning and enable high-performance control in unmapped environments. We present this solution in the context of simulated F1Tenth autonomous racing on unmapped tracks [6]. Fig. 1 shows how our approach uses the LiDAR scan to construct a representation of the visible environment that is used for optimisation-based racing. We evaluate our method with a two-layer optimisation and tracking strategy [7], and a single-layer Model Predictive Contouring Control (MPCC) algorithm [8]. We compare our methods to state-of-the-art mapless approaches of end-to-end neural networks [9] and the Follow-The-Gap (FTG) algorithm [2].

II. LITERATURE STUDY

Fig. 2 shows how autonomous racing methods can be split into classical methods, which use a perception and optimisation-based planning approach, and mapless methods, which use reactive control algorithms.

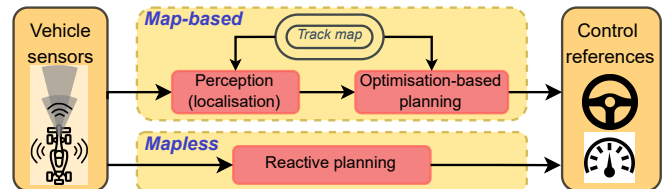


Fig. 2. Map-based planners using a perception and optimisation-based strategy compared with mapless methods.

A. Classical Racing

Classical racing methods use a perception, planning and control pipeline to move the vehicle around the track as

¹Electrical and Electronic Engineering Department, Stellenbosch University, Stellenbosch, 7600, South Africa. bdevans@sun.ac.za; hebrecht@sun.ac.za; wjordan@sun.ac.za

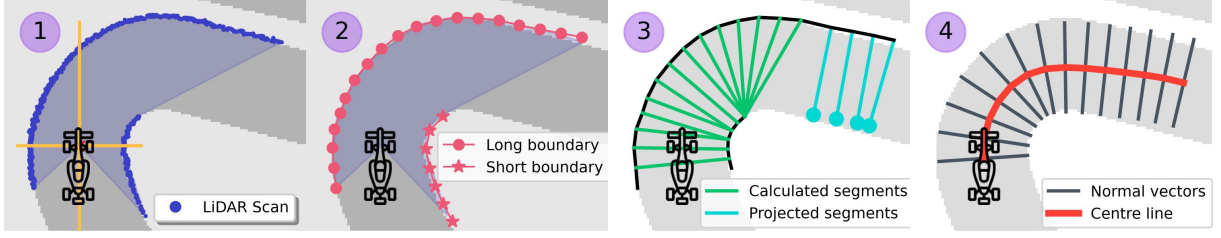


Fig. 3. Local map extraction: the track boundaries are identified and used to calculate a centre line and normal vectors of the visible region.

quickly as possible. Racing tracks are typically mapped using a Simultaneous Localisation And Mapping (SLAM) algorithm that requires a slow pass around the track to build a map [10]. During the drive through the map, the estimated odometry is fused with incoming LiDAR scans and/or camera images and used to build a map [11]. Localisation for autonomous racing is most commonly done with a scan-matching estimation algorithm that uses incoming odometry and sensor measurements to estimate the vehicle’s location [12], [13]. While localisation methods, such as particle filters, provide robust, accurate localisation, they are inherently limited by requiring a map of the race track. That makes them unsuitable for racing in unmapped or dynamically changing environments.

Classical racing typically uses a trajectory optimisation strategy to generate planning references of speed and steering angle. It is common to use a two-stage planning approach of calculating an optimal trajectory [7] and then following it with a path-following algorithm, such as pure pursuit [14], [15]. Another approach is to use a single-layer model predictive control (MPC) to directly optimise control commands that result in an optimal trajectory for a receding horizon [8], [16]. While optimal control strategies are effective for high-performance racing, they are limited by requiring a map of the track and the vehicle’s current location.

B. Mapless Racing

Mapless methods have been widely studied in the navigation literature, where the general problem is to move a holonomic robot from one point to another [17]. One of these methods, Follow-The-Gap [2], has been adapted (also known as the disparity extender algorithm) to autonomous racing where it has won a race [18]. However, these methods have no inherent model for speed selection and thus cannot operate the vehicle near the performance limits.

Another upcoming method is end-to-end deep learning, which uses a neural network to map raw sensor data directly to control commands. Bosello et al. [19] demonstrated that these methods can generalise to unseen race tracks. Current shortcomings in these methods are high-crash rates, even in simulation [20], the simulation-to-reality gap [21], and jerky action selection [9].

We aim to retain the high-performance nature of optimisation-based racing without requiring a global map. We do this by building a local map of the visible region that can be used for classical planning methods.

III. METHODOLOGY

We present the local map framework in the context of autonomous racing, where vehicles have a 2D LiDAR sensor for input and must select speed and steering actions that move the vehicle around the track as quickly as possible.

A. Local Map Extraction

The local map extraction uses the 2D LiDAR scan to build a set of centre line points and track widths that can be used for planning. Fig. 3 graphically illustrates this process, and Algorithm 1 provides pseudo code for the implementation.

Algorithm 1 Local map extraction algorithm

```

1: Receive LiDAR scan I
2:  $Z = \{[l_i \cos(\theta_i), l_i \sin(\theta_i)]\} \triangleright$  Scan to Cartesian points
3:  $Z \rightarrow B^{\text{long}}, B^{\text{short}} \triangleright$  Identify boundaries
4: Resample boundaries with  $n^{\text{long}}$  and  $n^{\text{short}}$  points
5:  $S = \emptyset \triangleright$  Initialise segment list
6: for  $i = 0, n^{\text{long}}$  do
7:    $\text{distances} = \{|\mathbf{b}_i^{\text{long}} - \mathbf{b}_j^{\text{short}}|_2 \mid j \in [0, n^{\text{short}}]\}$ 
8:    $k_i = \text{argmin}(\text{distances})$ 
9:   if  $|\mathbf{b}_i^{\text{long}} - \mathbf{b}_{k_i}^{\text{short}}|_2 < w_{\text{max}}$  then
10:     Append  $[\mathbf{b}_i^{\text{long}}, \mathbf{b}_{k_i}^{\text{short}}]$  to  $S$ 
11:   else
12:      $R^{\text{long}} \leftarrow B^{\text{long}}[i : n^{\text{long}}] \triangleright$  Remaining long line
13:      $R^{\text{short}} = R^{\text{long}} + \text{normals}(R^{\text{long}}) \times w_{\text{track}}$ 
14:     Append  $[R^{\text{long}}, R^{\text{short}}]$  to  $S$ 
15:     break out of for loop
16:   end if
17: end for
18: Use list  $S$  to calculate centre line and track widths

```

Fig 3 (1), shows the vehicle’s LiDAR scan as a set of points, defined by distances at set angles from the LiDAR scanner. Line 2 of the pseudo-code describes how they are converted to Cartesian points, Z , in the vehicle’s inertial frame, by multiplying by the corresponding angles θ_i . The LiDAR’s number of beams N_{beams} and field-of-view angle A are used to calculate the set of angles as,

$$\{-A/2 + n * A/N_{\text{beams}} \mid n \in [0, 1, \dots, N_{\text{beams}}]\}$$

. Fig 3 (2) shows how the set of points is split into left and right track boundary lines. Pseudocode lines 3 and 4 explain that the boundaries are treated in the categories of *long* and *short* and resampled to have equidistant points.

Lines 6-10 describe the progress to generate the green lines in Fig 3 (3). For each point in the long boundary $\mathbf{b}_i^{\text{long}}$, the distances to all the points on the short boundary are calculated (line 7). The nearest point $\mathbf{b}_{k_i}^{\text{short}}$ is found by taking the *argmin* of the distance array (line 8). If the distance between the points is smaller than the maximum track width w_{max} , then the segment is added to the segment list S (line 10). For the section where both boundaries are visible, this process finds track segments approximately normal to the track direction.

Lines 11-16 explain how the centre line is extended where only the longer boundary is visible. Line 12 identifies R^{long} as the long line section from the last segment added until the end of the line. The `normals` function returns the normal vectors indicating the direction across the track to the other boundary. The estimated short boundary R^{short} is then calculated in line 13 by adding the long boundary and the normal vectors multiplied by the track width w_{track} . The estimated segments R^{long} and R^{short} , shown as turquoise lines in Fig. 3 (3) are added to the segment list S .

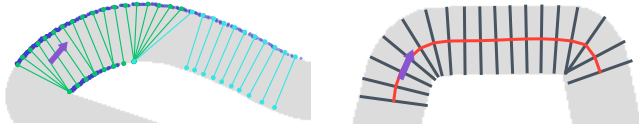


Fig. 4. Local map segment extraction (left) and a final local map (right) for segments of the AUT track. The purple arrow represents the car's pose.

The left image in Fig. 4 shows an example segment extraction, with the blue dots representing the LiDAR scan, the green lines representing the calculated segments, and the turquoise lines representing the projected segments. The green line segments ensure that the centre of the track is calculated since both boundaries are known. The turquoise line segments assume the track has a constant track width and that the borders are parallel to each other. The segment list is used to find the track centre line by adding finding the points in between the two boundaries. The line is then resampled to have equidistant points, and the track widths are calculated. The right image in Fig. 4 shows the resulting local map centre line and normal vectors that is returned and used for planning.

B. Optimisation Planners

We implement two standard planning techniques: a two-stage optimisation and tracking approach and a single-layer model predictive controller. These planning strategies are used with either global maps of the entire track and localisation or a local map.

1) *Two-stage Optimisation Planner*: Fig. 5 shows how the two-stage planner generates an optimal trajectory that is tracked using the pure pursuit algorithm. We use the optimisation approach presented by Heilmeyer et al. [7], generating a minimum curvature path and then a minimum time speed profile. The optimisation generates a path by minimising the curvature, which is defined as the rate of change in the heading. A forward-backwards solver calculates the speed

profile by selecting the fastest speed at each point that keeps the vehicle within the friction limits and is dynamically reachable.

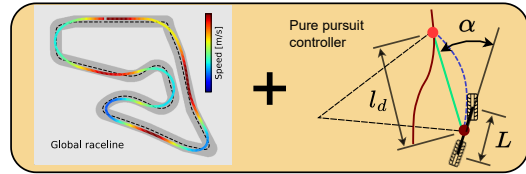


Fig. 5. The two-stage planner generates an optimal trajectory that is tracked with a pure pursuit controller

The pure pursuit formula [22] (shown on the right in Fig. 5) calculates a steering angle that tracks an upcoming waypoint that is lookahead distance l_d away, at an angle of α as,

$$\delta = \arctan\left(\frac{L \sin(\alpha)}{l_d}\right). \quad (1)$$

The lookahead distance is selected based on the speed as, $l_d = l_c + l_s \times V$ where l_c and l_s are hyperparameters and V is the vehicle speed.

2) *Model Predictive Contouring Control*: We adapt the Model Predictive Contouring Control (MPCC) algorithm presented by Liniger et al. [8] to F1Tenth racing. The optimisation represents the vehicle state as position and heading and the inputs as steering and speed for a finite number of steps. Successive states are constrained to the vehicle dynamics, where a kinematic bicycle model is used to update the states based on the control inputs. Additionally, the trajectory is constrained to lie within the track boundaries.

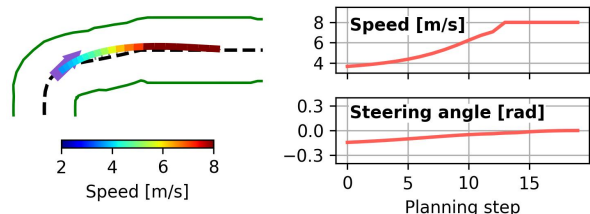


Fig. 6. The MPCC algorithm plans a receding horizon trajectory of speed and steering angle references that maximises centre-line progress.

Fig. 6 shows how the MPCC algorithm plans a finite trajectory by selecting speed and steering angle references. Since the nearest point on the reference path cannot be directly found, an approximate point on the centre line is used and added as an additional state to the optimisation variables. A lag error penalises the difference between the approximate centre line point and the true point along the trajectory. A contouring error encourages the trajectory to track the centre line. A progress objective promotes progress along the path, and a control regulation term encourages smooth control actions.

IV. EVALUATION

A. Methodology

The evaluation compares local map planners with global map planners and mapless methods. We implement the two optimisation strategies presented in Section III-B with the global and local planners. We use baseline mapless methods of end-to-end neural network controllers [9] and the FTG method [2]. We train end-to-end agents with the SAC and TD3 reinforcement learning algorithms using two hidden layers of 100 neurons, the trajectory-aided learning reward and 60,000 training steps on the GBR map.



Fig. 7. The AUT, ESP, and GBR (left to right) track maps.

We evaluate our method for F1Tenth autonomous racing using the simulator presented by O’Kelly et al. [6]. The vehicle is represented using the single-track bicycle model, and the dynamics equations are updated at 100 Hz. All the planners are run at 25 Hz, and at each planning step, a steering angle and linear speed must be selected to control the car. Fig. 7 shows the AUT, ESP and GBR maps used for simulation testing. We conduct the following tests:

- 1) **Local map extraction:** We investigate our pipeline’s ability to extract local maps
- 2) **Racing performance comparison:** We compare the lap times and speed profiles of the planners
- 3) **Computational requirements:** We measure the computation of each algorithm

B. Local Map Extraction

We investigate the local map extraction pipeline that uses the incoming 2D LiDAR scan to generate a local map. We measure the length of the centre lines of the extracted local maps for each track by using the pure pursuit algorithm to track the centre line at a low, constant speed.

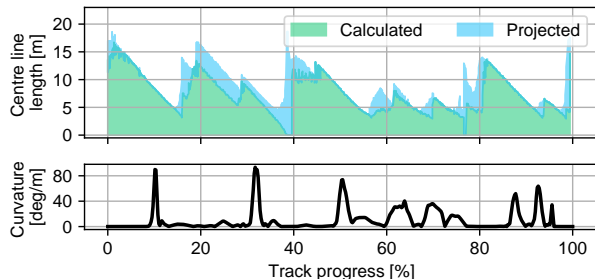


Fig. 8. Calculated and projected local map lengths compared to curvature.

Fig. 8 shows the calculated and projected local map lengths compared to curvature for a lap on the AUT track. The centre line lengths show a pattern of downward slopes followed by sharp rises. The curvature graph shows that the sharp rises appear after a curvature spike, indicating that the

Statistic	AUT	ESP	GBR
Mean \pm Std. dev.	11.05 \pm 4.52	11.10 \pm 4.76	11.03 \pm 4.77
Min, Max	3.44, 21.35	2.57, 25.96	2.64, 25.96

TABLE I

THE MEAN, STANDARD DEVIATION, MINIMUM AND MAXIMUM LOCAL MAP LENGTHS FOR THE AUT, ESP AND GBR TRACKS.

sharp rise is the straight that is suddenly visible after turning a corner. Table I shows that the maps have a mean centre line length of around 11 m, with a standard deviation of around 4.5 m. The local map lengths range between 2.57 m and 25.96 m. We conclude that the centre line can robustly be extracted and used for planning for the entirety of the track.

C. Racing Performance Comparison

We investigate the racing performance of the global and local optimisation planners, the Follow-The-Gap algorithm and the SAC and TD3 end-to-end agents. For each planner, five test laps are run with random start positions, and the average times from completed laps are presented.

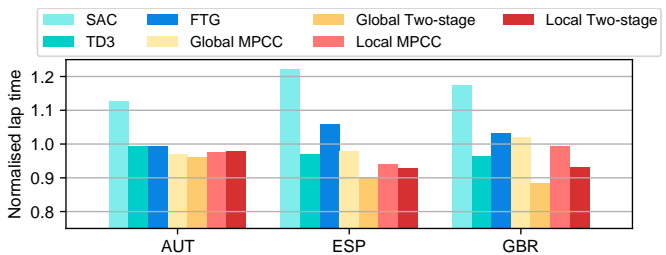


Fig. 9. Normalised lap times from the SAC and TD3 agents, FTG method, and global and local MPCC and two-stage planners.

Fig. 9 shows a bar graph of the lap times for each planner, normalised by dividing by the mean lap time for each track. The end-to-end agents and Follow-The-Gap method have the slowest times on all the maps. The global and local planners achieve fast times, with the two-stage planners outperforming the MPCC planners. Therefore, we study the performance of the two-stage planners, TD3 agent and Follow-The-Gap method in detail.

Map	TD3	FTG	Global	Local
AUT	19.13 (1.5%)	19.11 (1.4%)	18.52 (-1.8%)	18.85
ESP	41.95 (4.5%)	45.77 (14.1%)	38.97 (-2.9%)	40.13
GBR	36.71 (3.6%)	39.31 (11.0%)	33.58 (-5.2%)	35.42
Mean	3.22%	8.80%	-3.28%	-

TABLE II

LAP TIME IN SECONDS (% DIFFERENCE FROM THE LOCAL PLANNER) FOR THE TD3, FTG, AND GLOBAL AND LOCAL TWO-STAGE PLANNERS.

Table II shows the lap times and percentage difference from the local planner. The local map planner outperforms the TD3 agent by 3.22% and the Follow-The-Gap planner

by 8.8%. We investigate this result by plotting the trajectory segments on the ESP map in Fig. 10. The FTG planner takes a short, smooth path but has poor speed selection due to not using a model of the track. The end-to-end planner selects a more appropriate speed profile, but the trajectory is not smooth, and consequently, the vehicle selects conservative speeds. The local MPCC planning speeds up in the straights and slows down in the corners while selecting a smooth path. This fast performance outperforms other mapless methods.

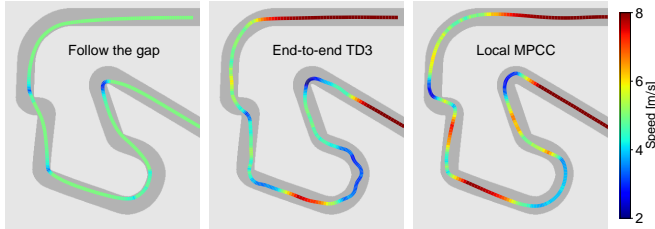


Fig. 10. Trajectory segments on the ESP map of the follow the gap, end-to-end and local MPCC planners.

Table II shows that the global two-stage planner achieves faster lap times on all the maps than the local two-stage planner. On average, the global planner lap times are 3.28% faster. We investigate this by plotting trajectory segments for a portion of the GBR map in Fig. 11. The two trajectories show a similar pattern of speeding up to high speeds in the straighter sections and slowing down enough for the corners, with the local planner slowing down more in the turns.

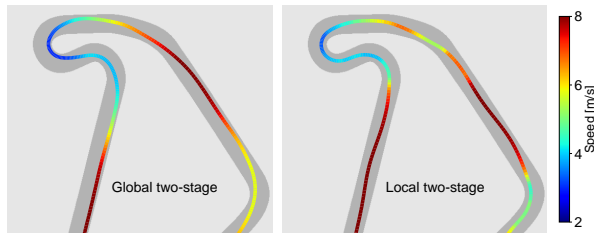


Fig. 11. Global and local two-stage planner trajectories on the GBR map.

Fig. 12 shows the speed profiles from the global and local two-stage planners on the AUT track. The global planner smoothly speeds up and slows down due to its ability to optimise the profile over the entire track. In contrast, the local planner shows more extreme behaviour of quickly speeding up to the maximum speed and slowing down to low speeds of around 2 m/s in the corners.

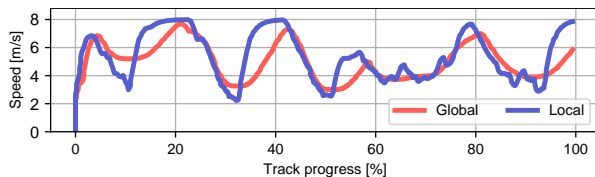


Fig. 12. Comparison of speed profiles from global and local two-stage planners on the AUT track.

Fig. 13 shows two example local racelines with the vehicle location represented by the purple arrow. These racelines

illustrate the limited visible planning horizon, which results in the local planner selecting a conservative trajectory.

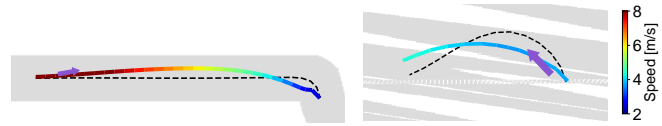


Fig. 13. Local racelines generated by the two-stage planner on AUT.

Our approach outperforms previous state-of-the-art approaches to autonomous racing of end-to-end neural networks and the Follow-The-Gap method by achieving lower lap times and smoother paths. The reason for this is that using local map representations of the visible track enables the use of an optimisation strategy. In comparison to global planning methods, our approach shows only a 3% performance drop. The drop is explained by the local map planner's conservatism around corners due to the lack of visibility. The local map planner selects a speed profile similar to the global planner's while using only the currently visible local map.

D. Computational Requirements

We investigate the perception and planning computation times of the racing algorithms. The perception refers to localisation using a particle filter for the global planners and local map generation for the local planners. All the tests are written in Python, run on an Intel i7-10700 desktop computer running Ubuntu 22.04 and use the cProfile library.

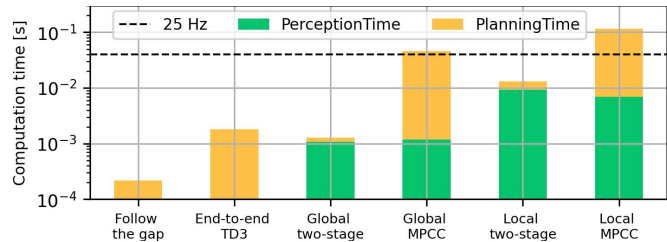


Fig. 14. Computation times for perception and planning on ESP.

Fig. 14 shows each planner's average perception and planning computation times. The Follow-The-Gap algorithm takes around 0.3 ms to plan, followed by the end-to-end agent taking around 1.2 ms. The global planners have perception (localisation) times of around 1 ms, and the local planners have longer perception (local map generation) times of around 10 ms. The two-stage planners run quickly, taking only 0.1 ms for the global and 0.3 ms for the local planner. The dashed black line shows the time required to run the algorithm at 25 Hz (40 ms). The global and local MPCC algorithms exceed the time constraint for real-time computation. The local map planner can run in real time if combined with an efficient algorithm.

V. CONCLUSION

This article presented the local map framework for optimisation-based control in unmapped environments. The

LiDAR scan was used to extract a local map of the visible area for F1Tenth autonomous racing. Our method extracted local maps that could be used for input to an optimisation strategy for planning. Local maps with an average of 11 m can be extracted from differently shaped racetracks. Our two-stage local map planner achieves lap times 3.22% faster than end-to-end agents trained with the TD3 algorithm and 8.8% faster than the Follow-The-Gap method. The primary improvement source is the local map planner's access to a vehicle dynamics model that can be optimised. The comparison with global planning approaches showed an average of 3.28% slower lap times, resulting from limited planning horizon around corners and thus reduced speed. Critically, our approach removes the limitation of global approaches in requiring a track map and localisation and, thus, enables high-performance racing on unmapped tracks.

A. Future Work

Image-based Control: Image-based control is difficult due to the large variability in images. Image-based mapping and localisation methods are computationally expensive, and end-to-end methods have not yet demonstrated satisfactory results [23]. However, the local map framework could use an edge detection algorithm to detect track boundaries and build a local map. Extracting low-level features can be used to enable high-performance image-based control.

Local Map Fusion for High-speed SLAM: Once a lap of a race track has been completed with the LocalMap planner, the local maps could be fused together to form a global map. Further, this can be extended to a full racing-optimised version of SLAM that builds a map in real-time by fusing successive local maps and estimating the difference. This will result in efficient, high-speed racing SLAM.

Safe Reinforcement Learning: Safe reinforcement learning, where the agent learns onboard a physical vehicle, has previously required a track map to build a kernel of safe states [24]. Previously, safe learning was limited to mapped contexts where localisation was available. Using local maps would enable a kernel of safe states to be built online, thus enabling safe learning to occur on unmapped tracks. Safe learning has the potential to achieve high-performance racing since it avoids the simulation-to-reality gap.

REFERENCES

- [1] A. Wischnewski, M. Geisslinger, J. Betz, T. Betz, F. Fent, A. Heilmeyer, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle *et al.*, "Indy autonomous challenge-autonomous race cars at the handling limits," in *12th International Munich Chassis Symposium 2021*. Springer, 2022, pp. 163–182.
- [2] 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.
- [3] 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.
- [4] S. A. Mohamed, M.-H. Haghbayan, T. Westerlund, J. Heikkonen, H. Tenhunen, and J. Plosila, "A survey on odometry for autonomous navigation systems," *IEEE access*, vol. 7, pp. 97 466–97 486, 2019.
- [5] 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.
- [6] 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.
- [7] A. Heilmeyer, 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.
- [8] 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.
- [9] 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.
- [10] F. Nobis, J. Betz, L. Hermansdorfer, and M. Lienkamp, "Autonomous racing: A comparison of slam algorithms for large scale outdoor environments," in *Proceedings of the 2019 3rd international conference on virtual and augmented reality simulations*, 2019, pp. 82–89.
- [11] L. Andresen, A. Brandemuehl, A. Honger, B. Kuan, N. Vödisch, H. Blum, V. Reijgwart, L. Bernreiter, L. Schaupp, J. J. Chung *et al.*, "Accurate mapping and planning for autonomous racing," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 4743–4749.
- [12] Stahl, Tim, Wischnewski, Alexander, Betz, Johannes, and Lienkamp, Markus, "Ros-based localization of a race vehicle at high-speed using lidar," *E3S Web Conf.*, vol. 95, p. 04002, 2019.
- [13] 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.
- [14] 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.
- [15] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, p. In Press.
- [16] V. Cataffo, G. Silano, L. Iannelli, V. Puig, and L. Glielmo, "A nonlinear model predictive control strategy for autonomous racing of scale vehicles," 2022.
- [17] K. Yan and B. Ma, "Mapless navigation based on 2d lidar in complex unknown environments," *Sensors*, vol. 20, no. 20, 2020.
- [18] "The "disparity extender" algorithm, and f1/tenth," <https://www.nathanotterness.com/2019/04/the-disparity-extender-algorithm-and.html>, accessed: 2023-11-13.
- [19] 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.
- [20] 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.
- [21] A. Murdoch, J. C. Schoeman, and H. W. Jordaan, "Partial end-to-end reinforcement learning for robustness against modelling error in autonomous racing," *arXiv preprint arXiv:2312.06406*, 2023.
- [22] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.
- [23] 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.
- [24] 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.