MPPI with Control Barrier Functions for F1/10: Robust Safety Under Real-World Uncertainty

Vaibhav Thakkar, Rhythm Chandak, Junfei Zhan, Ethan Yu
University of Pennsylvania, Philadelphia, USA
{vaithak, rhythm, zjf2024, ethanyu}@seas.upenn.edu

***Abstract*—This paper proposes a safety-critical control framework for autonomous racing on the F1TENTH platform, integrating Control Barrier Functions (CBFs) with the Model Predictive Path Integral (MPPI) method to achieve robust safety under real-world uncertainties. The Shield-MPPI controller enhances the traditional MPPI approach by embedding CBF-based safety constraints, enabling the vehicle to navigate complex racetracks while avoiding static and dynamic obstacles. Key components include real-time opponent detection using an Adaptive Breakpoint algorithm, opponent tracking via an Extended Kalman Filter (EKF), spline-based evasion planning for smooth trajectory generation, a state machine for adaptive decision-making and an MPPI based controller to follow the reference trajectory and enforcing control barrier function. Implemented using ROS2 for modularity and JAX for computational efficiency, the system operates at 50 Hz, meeting real-time requirements. Experimental evaluations in simulated and real-world environments demonstrate the capability of the framework to ensure safety, maintain accurate raceline tracking, and perform efficiently, offering a solution for safety-critical autonomous racing applications. Code repository link: https://github.com/vaithak/f1tenth_shield_mppi**

***Index Terms*—MPPI, Control Barrier Functions, Autonomous Racing, F1/10, Robust Control, Safe Planning**

I. Introduction

Autonomous racing is a demanding area for creating real-time motion planning and control systems. These systems need to drive fast on complicated racetracks while remaining safe and reliable. The F1TENTH platform, which is a small 1/10th scale autonomous racing car, is a standard way to test these systems under strict limits on computing power and how the car moves [1], [2]. In these situations, the vehicle must follow the planned path accurately, react quickly to moving obstacles, and handle problems such as noisy sensors, differences between the car model and reality, and changes in the environment [3], [4]. It is very important to incorporate safety features into these high-speed control methods to avoid crashes and keep the vehicle stable in these challenging conditions [5].

Among control methods that use sampling, model predictive path integral control (MPPI) has become a popular way to deal with complex car movements and difficult goals [6], [7]. MPPI works by trying out many possible paths and choosing the best ones using a method called importance sampling. This makes MPPI flexible and allows it to perform well in different tasks of robots, including driving aggressively [8]. However, MPPI does not always have clear safety rules, which can lead to dangerous situations when moving things, like other cars on a race track. Also, because MPPI needs to test many paths, it can take a lot of computing power, which is a problem for small computers like the ones used in the F1TENTH car.

To solve these problems, Control Barrier Functions (CBFs) offer a way to ensure that the car stays safe by keeping it within certain safe areas [5], [9]. CBFs give the guarantee that the car will remain in the safe zone, which is important for applications where safety is critical. Recently, people have been looking at combining CBFs with sampling-based planners to get both good performance and safety. An example is Shield-MPPI, which adds a CBF-based safety check into how MPPI chooses its paths. It also has a way to fix the planned path if it is not safe using some mathematical rules [10]. This combined approach maintains the good parts of MPPI for handling complex movements while also making sure that the path is safe and reliable [11]–[13].

Obstacle avoidance is a critical component in autonomous racing, where the vehicle must navigate safely around static and dynamic obstacles. Static obstacles, such as track boundaries or debris, require precise path planning to maintain safety, while dynamic obstacles, such as opponent vehicles, require real-time reactivity and prediction to avoid collisions [14], [15]. Traditional methods often address these challenges separately, using techniques such as potential fields for static obstacles and predictive models for dynamic ones [16], [17]. However, integrating both types of obstacle avoidance into a single real-time framework remains a significant challenge, particularly under the computational constraints of platforms such as F1TENTH. Our work addresses this by incorporating both static and dynamic obstacle avoidance into the Shield-MPPI framework, utilizing the Extended Kalman Filter (EKF) for opponent tracking, the Spline-Based Evasion Planner for smooth and feasible avoidance trajectories, and a State Machine for adaptive decision-making based on obstacle type and proximity, all while enforcing safety constraints through CBFs.

This paper presents the application of Shield MPPI to autonomous racing on the F1TENTH platform, with the following contributions:

- Implementation of the Shield-MPPI controller on the F1TENTH platform, integrating CBFs to enforce safety while maintaining aggressive driving capabilities.
- Development of a multilayer architecture comprising opponent tracking, spline-based evasion planning, adaptive state management, and MPPI control.

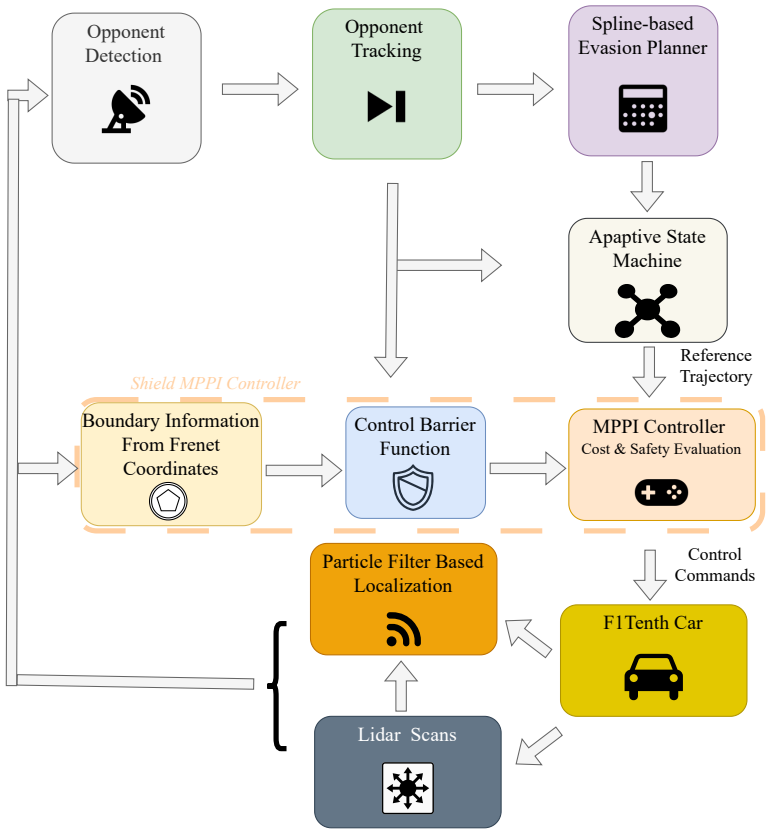


Fig. 1: Overview of the control framework for autonomous racing on the F1TENTH platform.

- Real-time system implementation using ROS2 for modular communication and JAX for computational efficiency, achieving a 50 Hz control frequency.
- Comprehensive evaluation of the proposed framework in simulated and real-world racing scenarios, demonstrating the effectiveness of CBF-based safety constraints in dynamic obstacle interactions.

II. Related Work

Our research overlaps with work that aims to develop motion planning and control systems for autonomous racing, advance sampling-based control methods such as MPPI, and ensure safety through control barrier functions. This section reviews prior work in these areas, providing context for our implementation of Shield-MPPI on the F1TENTH platform.

A. Autonomous Racing and Motion Planning

Autonomous racing demands real-time motion planning and control systems that balance performance and safety in dynamic environments. The F1TENTH platform has become a widely adopted benchmark for testing such systems under computational and dynamical constraints [1], [2]. For example, O'Kelly et al. [3] developed a scalable perception-aware planning framework for autonomous racing, addressing real-time computation and sensor noise robustness. Similarly, Baumann et al. [18] presented the ForzaETH race stack, integrating perception, planning, and control for high-speed racing. These studies emphasize the need for efficient algorithms that can handle complex racetracks and dynamic obstacles, laying the groundwork for our safety-focused racing approach.

B. MPPI Control

MPPI control is a sampling-based method well-suited for nonlinear dynamics and complex objectives in robotics. Williams et al. [6] introduced MPPI as an information-theoretic approach to model predictive control, demonstrating its efficacy in autonomous driving. Its ability to explore control sequences via importance sampling has been applied to aggressive driving scenarios [8] and general robotic tasks [7]. However, MPPI's lack of inherent safety guarantees and high computational requirements limit its use in safety-critical settings. Enhancements like robust MPPI [12] offer performance guarantees under uncertainty, while risk-aware MPPI [11] uses conditional value-at-risk to manage cost uncertainty. Furthermore, Yin et al. [13] proposed trajectory distribution control through covariance steering, improving MPPI's sampling efficiency.

C. CBFs for Safety

CBFs provide a formal method to enforce safety by constraining the trajectories of the system within safe sets. Ames et al. [5] applied CBFs to safety-critical systems, integrating them with quadratic programs for real-time control, while [9] showcased their use in adaptive cruise control. In robotics, CBFs have been used for collision avoidance in multi-agent scenarios [19]. Combining CBF with sampling-based methods such as MPPI is challenging due to the stochastic nature of MPPI, but Yin et al. [10] introduced Shield-MPPI, which embeds CBF penalties into MPPI's cost function and uses constrained optimization to ensure safety. This approach inspires our work, which enables safe and efficient control in autonomous racing.

III. Methodology

Fig. 1 illustrates the control framework for autonomous racing on the F1TENTH platform, highlighting the systematic flow of information across key computational modules to enforce safety and optimize control through CBF. The framework begins with opponent detection and particle filter-based localization, integrating sensor data to inform the adaptive state machine. The state machine governs the operational mode based on the CBF constraints, the information on the trajectory, and the environmental context. Moreover, the spline-based evasion planner generates feasible paths around detected obstacles, feeding these to the MPPI controller for cost and safety evaluation. The MPPI controller assesses multiple trajectories, applying control barrier function penalties to ensure safety compliance. The refined control commands are then executed by the F1Tenth car, closing the feedback loop and enabling robust real-time navigation in dynamic racing scenarios. Further implementation details are elaborated in subsequent sections.

A. Opponent Detection

The opponent detection module is designed to identify and track obstacles along the racetrack using LiDAR data and vehicle odometry [20]. This module processes raw sensor data to extract potential obstacles, segment them into distinct clusters, and estimate key properties such as position, size, and orientation.

Initially, the module transforms LiDAR scan data from the vehicle frame to a global map frame using a transformation matrix:

$$H_{l2m} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where x and y are the global coordinates of the vehicle, and θ is the orientation of the vehicle obtained from odometry.

The transformed data is then segmented to detect clusters of points that likely represent obstacles. This segmentation employs an adaptive thresholding method to dynamically determine the maximum allowable distance between consecutive points based on the current range and angular increment:

$$d_{max} = r \cdot \frac{\sin(\lambda - \Delta\phi)}{\sin(\Delta\phi)} + 3\sigma \tag{2}$$

where r is the range to the point, λ is the predefined angular resolution, $\Delta\phi$ is the angular increase, and σ is the standard deviation of sensor noise.

After segmentation, identified clusters are analyzed to fit bounding boxes, allowing calculation of obstacle dimensions and orientations. The center of each obstacle is then projected onto the Frenet coordinate system of the track to facilitate trajectory planning and control.

This structured approach enables the detection of dynamic and static obstacles in real time, providing essential information for subsequent trajectory planning and decision-making modules.

B. Opponent Tracking

The opponent tracking module processes sensor data to detect and monitor dynamic obstacles, such as opponent vehicles, in real time. Utilizing an EKF, this subsystem estimates the opponent's state in Frenet coordinates relative to the raceline [18]. The state vector is defined as:

$$\mathbf{x} = \begin{bmatrix} s & v_s & d & v_d \end{bmatrix}^T \tag{3}$$

where s represents the longitudinal position along the track, v_s is the longitudinal velocity, d is the lateral offset from the raceline, and v_d is the lateral velocity.

According to Fig. 2, the Frenet coordinate system offers a curvilinear reference frame that decomposes the vehicle's state into longitudinal and lateral components relative to the raceline. This decomposition simplifies motion estimation along complex, non-linear trajectories by aligning positional and velocity vectors with the curvature of the track, thereby enhancing the tracking accuracy of opponent vehicles.

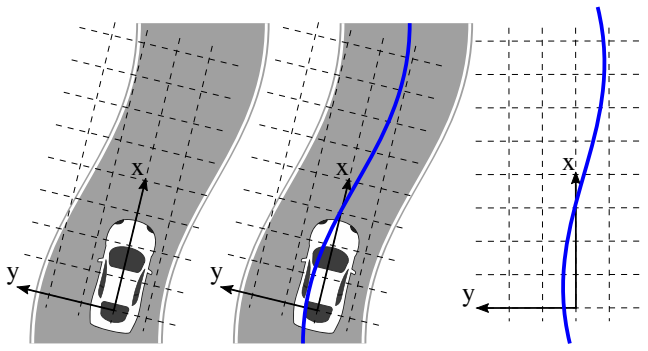


Fig. 2: Frenet coordinate system representing the raceline, with longitudinal s and lateral d components [21].

The EKF prediction step employs a constant velocity model with proportional damping on lateral states to stabilize estimates:

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \tag{4}$$

where the state-transition matrix $\mathbf{F}$ is:

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

and $\mathbf{u}_k = \begin{bmatrix} 0 & 0 & -P_d d_k & -P_{v_d} v_{d_k} \end{bmatrix}^T$ applies damping with gains P_d and P_{v_d}. The update step incorporates sensor measurements to refine the state estimate, ensuring accurate tracking.

C. Spline-Based Evasion Planner

To enable safe and efficient overtaking during autonomous racing, a spline-based evasion planner is activated when an opponent vehicle is detected within a predefined lookahead distance $l_a = 7.0\,\mathrm{m}$. The planner generates a smooth trajectory that deviates from the raceline to safely bypass the obstacle while ensuring compliance with track boundaries and vehicle dynamics [18].

Obstacle filtering is performed first by evaluating the lateral deviation d of each detected object from the raceline and its longitudinal position s relative to the ego vehicle. An obstacle is considered relevant if it satisfies

$$|d| < \theta \quad \text{and} \quad \Delta s = (s_{\text{obs}} - s_{\text{ego}}) \bmod L < l_a, \tag{6}$$

where $\theta = 0.35\,\mathrm{m}$, L is the total track length, and modular arithmetic ensures proper handling of wrap-around positions.

For dynamic obstacles, future locations are predicted using a constant velocity model. The planner then determines the preferred overtaking side by comparing the available lateral gaps g_ℓ and g_r on the left and right. An overtaking direction is selected if the corresponding gap exceeds the required clearance

$$g > d_e + \delta, \tag{7}$$

where $d_e = 0.3\,\mathrm{m}$ is the target evasion distance and δ is a safety buffer. If both sides are viable, the planner favors the side closer to the raceline or the outer side of the upcoming turn, based on curvature integration over a finite horizon.

A set of offset waypoints (s_i, d_i) is then constructed around the obstacle's center s_a, with longitudinal spacing scaled by vehicle speed to preserve temporal resolution. The lateral offsets d_i form a trapezoidal profile centered at s_a, peaking at d_e near the obstacle and decaying to zero away from it.

A cubic spline is fitted to the set $\{(s_i, d_i)\}$, yielding a continuous deviation function

$$d(s) = \text{Spline}(s_i, d_i). \tag{8}$$

The spline is discretized at fixed longitudinal intervals, and each point d_j is clipped to ensure it remains within bounds defined by the selected overtaking side.

To validate the planned path, each point is checked against the local track width constraint:

$$|d_j| \leq w_j - \delta, \tag{9}$$

where w_j is the available lateral space at position s_j. If this constraint is violated at any point, the trajectory is discarded. Otherwise, the valid spline is converted from Frenet to Cartesian coordinates for downstream tracking and control.

D. State Machine

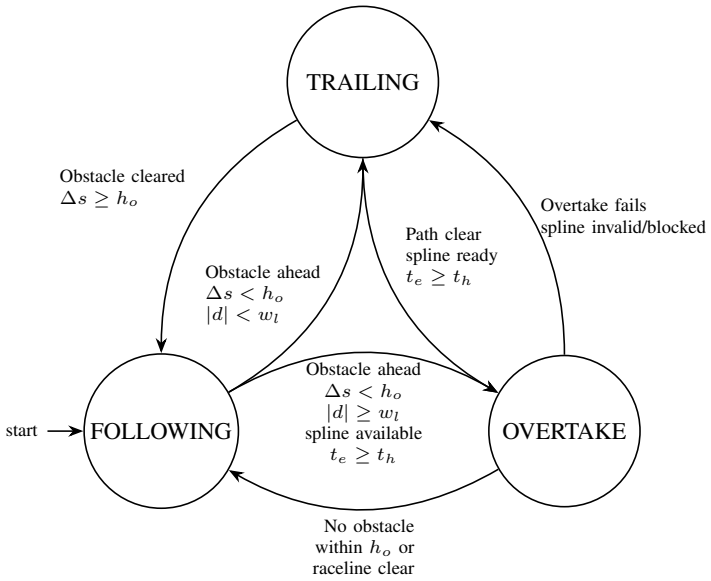


Fig. 3: High-level state machine for race-line following, overtaking, and trailing. Adapted from ForzaETH race stack.

The state machine in Figure 3 governs high-level decision making by orchestrating transitions between distinct behavioral modes, allowing the vehicle to adapt its strategy to real-time racing conditions. The three primary operational states include the following, overtaking, and trailing of the race line. In the following state of the race line, the vehicle follows a predefined global (raceline) trajectory optimized for lap time. When a slower opponent is detected ahead, the vehicle may transition to the overtaking state, during which a dynamically generated evasion trajectory is executed. If overtaking is not immediately feasible, the vehicle enters a trailing state, maintaining a safe distance behind the opponent until conditions permit a maneuver.

The transitions between these states are governed by a rule-based system that merges information about the environment, obstacle configuration, and vehicle status. One key factor is the proximity of the obstacle. The state machine monitors all detected obstacles within an overtaking horizon of $h_o = 6.0\,\mathrm{m}$, computing their longitudinal distance in Frenet coordinates:

$$\Delta s = (s_{\text{obs}} - s_{\text{ego}}) \bmod L, \tag{10}$$

where s_{obs} and s_{ego} denote the obstacle and ego vehicle longitudinal positions, and L is the track length. An obstacle is considered within the overtaking zone if $\Delta s < h_o$. To assess whether the race line is blocked, the lateral distance $|d|$ of each

obstacle from the centerline of the track is evaluated. The path is considered clear if $|d| \geq w_l$, where $w_l = 0.3\,\mathrm{m}$ is the lateral blocking threshold.

Trajectory availability is another critical condition. A state transition into overtaking is only allowed if a valid spline-based evasion trajectory is available. To prevent erratic switching, a hysteresis timer is used. Let $t_e = t_c - t_s$, where t_c is the current time and t_s is the timestamp of the last transition. A new state is activated only when $t_e \geq t_h$, with $t_h = 0.3\,\mathrm{s}$ denoting the minimum hold time.

Finally, some transitions require the vehicle to be sufficiently aligned with the raceline. This is enforced by checking the ego vehicle's lateral deviation d_c, which must satisfy

$$|d_c| < w_e, \tag{11}$$

where $w_e = 0.15\,\mathrm{m}$ is a positional tolerance threshold.

These decision criteria are modular and mode-dependent, allowing flexible adaptation to different racing formats such as time trials or competitive duels. Operating at 50 Hz, the state machine ensures fast and robust switching, facilitating seamless integration between global planning and local evasion strategies.

E. Shield MPPI Controller

The Shield MPPI controller is a safety-aware trajectory planner that integrates MPPI control with Discrete-Time Control Barrier Functions (DCBFs) to generate optimal and constraint-satisfying control inputs for autonomous racing vehicles [10]. MPPI is a sampling-based stochastic optimal control method that evaluates multiple control trajectories over a finite horizon by minimizing a predefined cost functional. In this framework, safety constraints are enforced through DCBFs, which ensure forward invariance of a safe set and provide formal guarantees for obstacle avoidance and adherence to track boundaries.

At each planning step, the controller samples a set of candidate control sequences. Each trajectory is evaluated using an augmented cost function of the form:

$$J(\mathbf{u}) = \mathbb{E}\left[\phi(\mathbf{x}_K) + \sum_{k=0}^{K-1}\left(q(\mathbf{x}_k) + \frac{\lambda}{2}\mathbf{u}_k^\top \Sigma^{-1}\mathbf{u}_k\right)\right], \tag{12}$$

where $\phi(\mathbf{x}_K)$ is the terminal cost, $q(\mathbf{x}_k)$ is the state-dependent running cost, λ is the temperature parameter, and Σ is the control noise covariance matrix.

The DCBF penalty term C_{cbf} is defined as:

$$C_{cbf}(x_k, x_{k-1}) = C\max\left\{-h(x_k) + \alpha h(x_{k-1}), 0\right\}, \tag{13}$$

where C is a penalty scaling parameter, $\alpha = 1-\beta \in (0,1)$, and $h(x)$ defines the safety constraint function. The DCBF function ensures that the state remains within a predefined safe set $\mathcal{S} \subseteq \mathbb{R}^n$, defined as:

$$\mathcal{S} = \{x \in \mathbb{R}^n \mid h(x) \geq 0\}. \tag{14}$$

To incorporate the DCBF term into the MPPI cost function, we augment the state to include both the current and previous states:

$$\mathbf{z}_k = \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix}, \quad \mathbf{z}_{k+1} = \begin{bmatrix} f(x_k, u_k) \\ x_k \end{bmatrix}. \tag{15}$$

The DCBF cost in the augmented system becomes:

$$C_{cbf}(\mathbf{z}_k) = C\max\left\{-h(z_k^{(1)}) + \alpha h(z_k^{(2)}), 0\right\}. \tag{16}$$

Thus, the modified trajectory cost for the m-th sample becomes:

$$\begin{aligned} S_m = \phi(z_{m,K}) + \sum_{k=0}^{K-1}\Big(&q(z_{m,k}) \\ &+ \lambda \mathbf{u}_k^\top \Sigma^{-1}\mathbf{u}_k + C_{cbf}(z_{m,k})\Big). \end{aligned} \tag{17}$$

If the nominal MPPI-generated trajectory violates the safety constraint, a local repair procedure is invoked to refine the control sequence. This repair step minimizes the DCBF violation term over a shorter horizon N as follows:

$$\mathbf{u}^* = \arg\min_{\mathbf{u}\in\mathcal{U}} \sum_{k=0}^{N-1} \min\left\{h(f(x_k, u_k)) - \alpha h(x_k), 0\right\}. \tag{18}$$

This optimization is performed using a gradient-based method to ensure rapid convergence under real-time constraints.

The resulting control inputs are then executed with saturation limits, maintaining feasibility under vehicle actuation capabilities. This hierarchical integration of sampling-based planning and constraint enforcement ensures robust, collision-free control under uncertainty.

F. Integration and Real-Time Operation

The complete control framework is implemented as a modular architecture within ROS2 middleware, enabling robust communication between subsystems and real-time responsiveness. Each component, opponent tracking, state management, spline planning, and Shield MPPI control, operates as an independent node, facilitating asynchronous processing and system scalability.

The opponent tracking module runs at 50 Hz to ensure timely updates on dynamic obstacles. The Shield-MPPI controller leverages JAX-based numerical optimization for efficient trajectory sampling and repair, operating within the real-time constraints of the F1TENTH platform. Evasion trajectories generated by the spline planner are seamlessly integrated into the decision-making pipeline via the state machine, enabling adaptive behavior in competitive settings.

As shown in Fig. 1, this integration forms a closed-loop system that continuously processes sensory input, updates internal state estimates, and executes safe, performance-oriented control actions in real time. This architecture enables the vehicle to navigate complex racing scenarios with both agility and safety guarantees.

IV. Evaluation

A. Experimental set-up

All experiments were performed on the standard F1TENTH 1/10-scale race-car and its companion Gazebo simulation. The vehicle runs a Jetson Orin Nano. The software stack executes as independent ROS2 nodes (Sec. III); their nominal update rates during the experiment are:

- State Machine: **50 Hz** timer (`rate_hz`)
- Spline-based Evasion Planner: **40 Hz** timer (`publish_rate`)
- Opponent Tracking EKF: **40 Hz** timer (`rate`)
- Shield-MPPI Controller: pose-triggered, effective ≈ 50 Hz (10 cm horizon with $\Delta t = 0.1$ s)

All nodes exchange data over DDS with `RELIABLE` QoS, ensuring deterministic message delivery at racing speeds.

B. Qualitative results

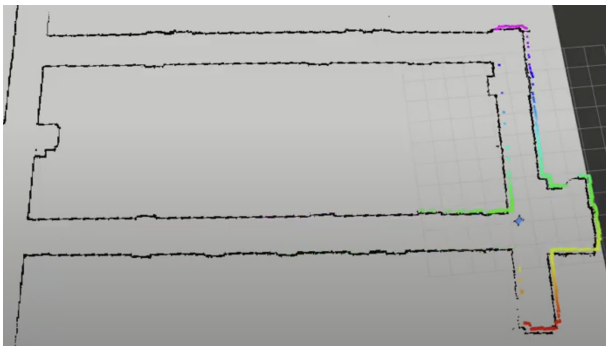
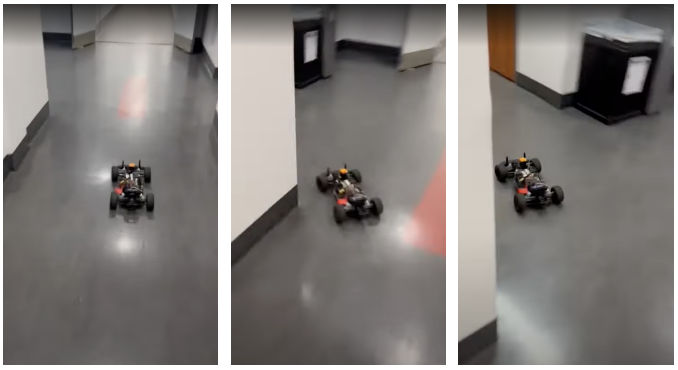


Fig. 4: Shield-MPPI following the global raceline in Gazebo. The ego vehicle (red) stays within the CBF-defined safe set while minimizing lap time.



(a) $t = 0$ s (b) $t = 0.7$ s (c) $t = 1.4$ s

Fig. 5: Three consecutive frames from the real-world run on the Penn F1TENTH track. Shield-MPPI tracks the raceline while respecting CBF safety constraints.

Figure 4 demonstrates the Shield-MPPI controller in simulation, while Fig. 5 shows the corresponding hardware experiment.

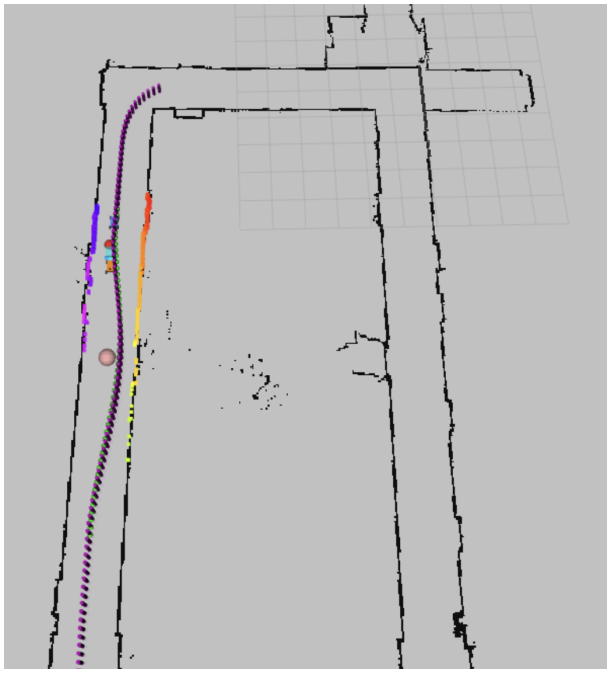
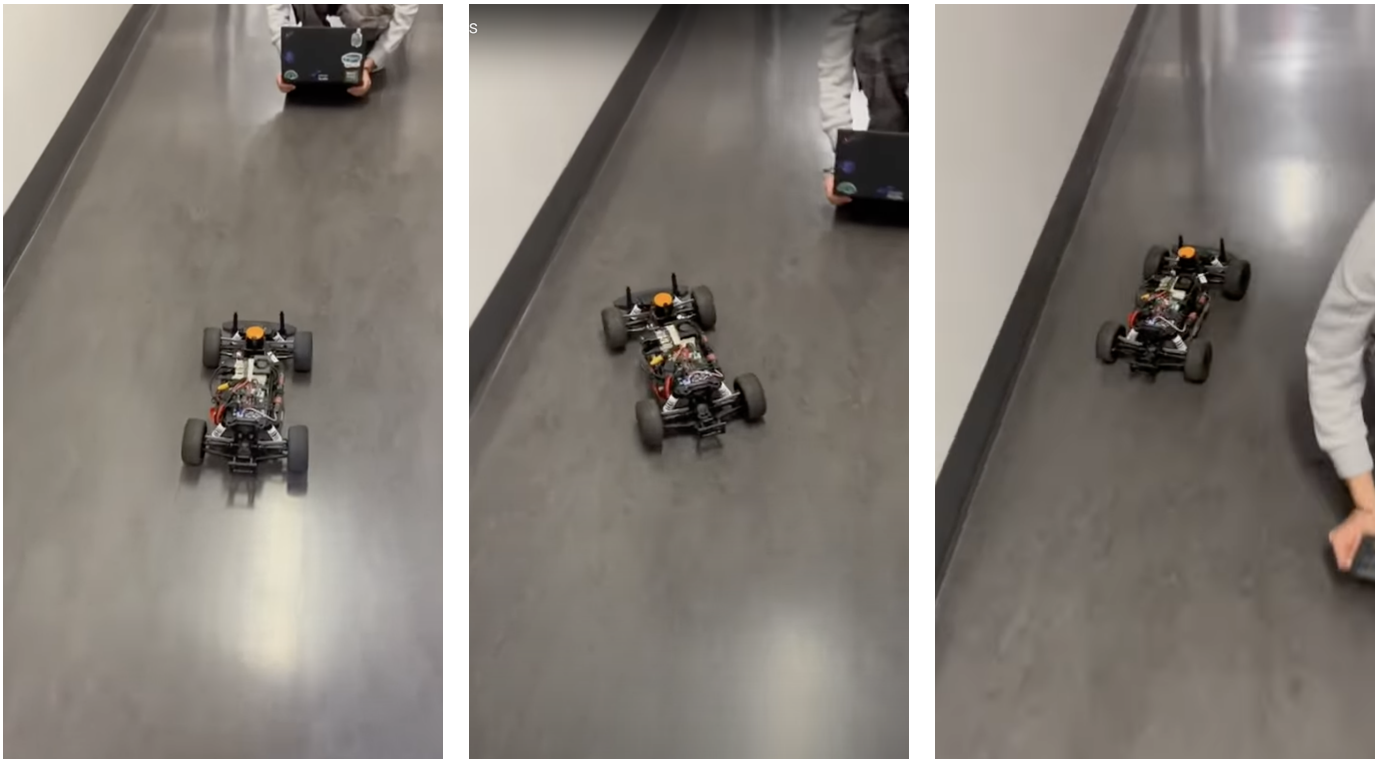


Fig. 6: Dynamic spline (blue) generated to bypass a tracked opponent (red). The spline is fed to Shield-MPPI as a reference trajectory.

Fig. 7: LiDAR-based opponent detection in simulation.



(a) The F1TENTH car, following the raceline trajectory, detecting an obstacle.

(b) Dyanmic spline generation and initiation of avoidance manuever.

(c) Successful obstacle avoidance as the car safely maneuvers around the detected object.

Fig. 8: Obstacle avoidance demonstration using the spliner-based evasion planner. The vehicle first detects the obstacle (a), adjusts its trajectory to avoid it (b), and then returns to the raceline (c).

Additional qualitative results are given in Fig. 7 (opponent detection), Fig. 6 (spline generation), and the obstacle-avoidance sequence in Fig. 8. Accompanying videos are:

- *MPPI in simulation:* https://www.youtube.com/watch?v=kFjm8tqywcs
- *MPPI on hardware:* https://www.youtube.com/watch?v=rvTRtGZ7j6I
- *Opponent detection (sim):* https://www.youtube.com/watch?v=4_vBc8fX8uw
- *Obstacle avoidance with Shield MPPI (sim):* https://www.youtube.com/watch?v=nVThgGBrOw4
- *Obstacle avoidance with Shield-MPPI (real):* https://www.youtube.com/watch?v=V5UYKdPNC0Q

C. Obstacle-avoidance pipeline.

The full pipeline—LiDAR-based detection $\rightarrow$ opponent tracking $\rightarrow$ spline generation $\rightarrow$ Shield-MPPI tracking—runs in real time on both simulation and hardware. Figure 8 captures a typical avoidance maneuver: after the obstacle is detected the spline planner generates an alternate path within 25 ms (one planning cycle). Shield-MPPI accepts the spline as a reference, samples 500 candidate control sequences, and outputs admissible steering/throttle commands that keep the vehicle within the CBF-defined safe set.

D. Runtime performance

Across ten one-lap trials (length 50 m) the stack maintained its nominal loop rates with no deadline misses. End-to-end latency—from LiDAR scan timestamp to actuator command—remained below 40 ms on hardware. No collisions or track-bound violations were observed; every detected obstacle (static cardboard box, 0.22 m cube) was safely avoided with a lateral clearance > 0.1 m.

Because this work focuses on demonstrating feasibility, no ablation study or quantitative lap-time benchmarking was conducted. Future work (Sec. V) will address these limitations.

V. Conclusion

This paper presented the first *Shield-MPPI* implementation on the F1TENTH platform, seamlessly blending Model Predictive Path Integral control with discrete–time Control Barrier Functions. The resulting controller preserves the aggressive performance of MPPI while enforcing provable safety constraints derived from CBFs. Key take-aways are:

1) A fully-integrated perception–planning–control pipeline running at 40–50 Hz on on-board compute, without GPU-exclusive operations.
2) Successful closed-loop obstacle detection, dynamic spline generation, and CBF-shaped trajectory tracking in both simulation and on the physical F1TENTH car.

3) A modular ROS2 implementation that the community can readily extend for multi-car racing scenarios.

Limitations and future work. The current evaluation is qualitative and restricted to single-opponent scenarios at low speed. Immediate next steps include (i) a quantitative study of lap times, energy efficiency, and safety-margin statistics; (ii) a systematic ablation of CBF penalties versus MPPI cost tuning; and (iii) extension to multi-opponent racing with negotiation of overlapping avoidance splines.

Overall, Shield-MPPI demonstrates that sampling-based optimal control and control-theoretic safety certificates can coexist in a real-time autonomous-racing stack, providing a promising foundation for future high-speed, safety-critical robotics applications.

REFERENCES

[1] S. Purwin, C. Best, U. Rosolia, T. Wu, and S. Karaman, "F1tenth autonomous racing: A benchmark," in *Proceedings of the 4th Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. J. Chung, F. Duvallet, M. J. Kochenderfer, D. Kulic, S. D. Oberdiek, A. Paraschos, M. Schwager, D. Song, and M. Toussaint, Eds., vol. 155, 2021, pp. 1037–1047.

[2] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," in *Proc. of the NeurIPS 2020 Competition and Demonstration Track*, ser. Proceedings of Machine Learning Research, Y. Deng, H. Meunier, A. Oliver, G. Farquhar, N. Novack, and Y. Wang, Eds., vol. 133. PMLR, 2021, pp. 77–89.

[3] M. O'Kelly, H. Zheng, K. Sycara, W. Schwarting, S. Karaman, and D. Rus, "Scalable perception-aware planning for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1092–1099, 2020.

[4] A. Brunnbauer, J. Berberich, M. Lechner, C. Mastalli, and M. Althoff, "Latent imagination facilitates zero-shot transfer in autonomous racing," *arXiv preprint arXiv:2210.00809*, 2022. [Online]. Available: https://arxiv.org/abs/2103.04909

[5] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 8, pp. 2920–2935, 2019.

[6] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[7] G. Williams, A. Aldrich, and E. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[8] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1433–1440.

[9] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 6271–6278.

[10] J. Yin, C. Dawson, C. Fan, and P. Tsiotras, "Shield model predictive path integral: A computationally efficient robust mpc approach using control barrier functions," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7106–7113, 2023.

[11] J. Yin, Z. Zhang, and P. Tsiotras, "Risk-aware model predictive path integral control using conditional value-at-risk," *arXiv preprint arXiv:2209.12842*, 2022. [Online]. Available: https://arxiv.org/abs/2209.12842

[12] M. S. Gandhi, B. Vlahov, J. Gibson, G. Williams, and E. A. Theodorou, "Robust model predictive path integral control: Analysis and performance guarantees," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1423–1430, 2021.

[13] J. Yin, Z. Zhang, E. Theodorou, and P. Tsiotras, "Trajectory distribution control for model predictive path integral control using covariance steering," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1478–1484.

[14] 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.

[15] 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, 2020.

[16] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[17] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[18] N. Baumann, E. Ghignone, J. Kühne, N. Bastuck, J. Becker, N. Imholz, T. Kränzlin, T. Y. Lim, M. Lötscher, L. Schwarzenbach, L. Tognoni, C. Vogt, A. Carron, and M. Magno, "Forzaeth race stack: Scalable ultra-high speed autonomous racing," *arXiv preprint arXiv:2403.11784*, 2024.

[19] M. Ahmadi, A. Singletary, J. W. Burdick, and A. D. Ames, "Safe policy synthesis in multi-agent pomdps via discrete-time barrier functions," in *IEEE Conference on Decision and Control (CDC)*, 2019, pp. 4797–4803.

[20] S. Vaskov, "Fast and safe trajectory optimization for autonomous mobile robots using reachability analysis," 2021. [Online]. Available: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/169729/skvaskov_1.pdf?sequence=1

[21] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 987–993.