OSCAR: Overtaking Strategy in Competitive Autonomous Racing

Tristan Scheiner*, Yongjun Cho*, Jiyeong Oh*
*Luddy School of Informatics, Computing, and Engineering,
Indiana University Bloomington, IN, USA
Emails: {trcschei, yc134, ohjiye}@iu.edu

***Abstract*—This paper presents OSCAR, a framework for competitive autonomous racing that combines Model Predictive Contouring Control (MPCC) with opponent-aware behavior prediction. MPCC enables high-speed, flexible path following with collision avoidance via soft constraints, while opponent behavior is modeled using information-theoretic bounded rationality. The approach is evaluated in low- and high-fidelity simulations and on a real F1TENTH vehicle, demonstrating robust performance, effective multi-vehicle interaction, and reliable simulation-to-real transfer.**

***Index Terms*—Model Predictive Control, Bounded Rationality, F1Tenth, ROS2, Autonomous Racing, Multi-Robot System**

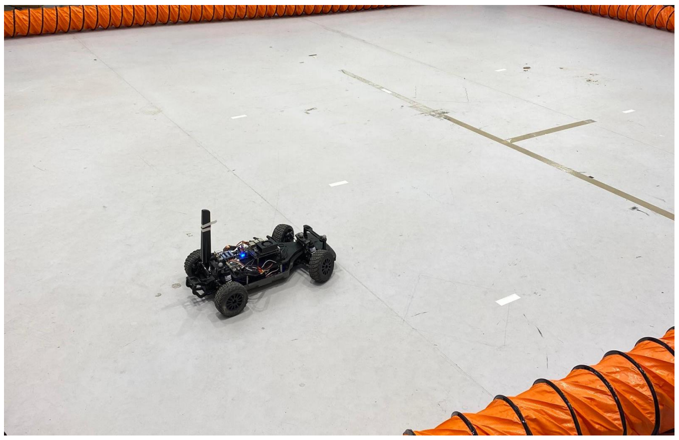
Fig. 1. Experimental setup on the custom indoor circuit. The F1TENTH vehicle is shown navigating within the track boundaries defined by flexible orange tubing, utilizing the Vicon motion capture system for real-time state estimation and MPCC controller with ROS2.

I. Introduction

A. Background

This project integrates high-performance control theory with the specific hardware to solve the autonomous racing problem. We utilize the F1TENTH research platform to validate a Bounded Rationality and a Model Predictive Contouring Control (MPCC) scheme. The specific components of this framework are defined as follows:

1) The F1TENTH Platform: We utilize F1TENTH, a 1:10 scale autonomous racing vehicle with Ackerman steering, designed for high-fidelity simulation research. This platform mimics the challenges faced by real-world autonomous vehicles by limiting the system to only use limited embedded computing (NVIDIA Jetson). There is potential for a variety of sensors in order to perceive the environment, for instance LiDAR and camera. This hardware configuration imposes strict latency requirements, making lightweight and robust control algorithms essential for real-world vehicle dynamics.

2) Bounded Rationality in Decision Making: Our planning architecture is based on the concept of bounded rationality, which assumes that rational decision-making is limited by the available information, computational resources, and time constraints required for decision-making. The Purpose of decision making is that an agent picks an actions that maximizes some utility. However, the bounded rationality models decision making where an agent satisfies some goal rather than strictly maximizes

3) From Model Predictive Control (MPC) to MPCC: We use MPCC to control the F1TENTH platform. The MPC effectively handles constraints by optimizing inputs over a finite time horizon, but it typically tracks a fixed time-parameterized reference trajectory. This approach reveals vulnerabilities in racing scenarios; if the vehicle falls behind the reference time, a typical MPC will aggressively attempt to "catch up," often leading to instability. To address this issue, we used MPCC. Instead of tracking a specific point in time, MPCC introduces a progress variable (θ) to track the geometric path itself. The controller minimizes the lateral contour error while simultaneously maximizing the progress speed along the track, allowing the vehicle to dynamically adjust its speed to match cornering limits without being constrained by a pre-calculated speed profile.

B. Motivation & Objectives

Autonomous racing provides a challenging testbed for decision-making under dynamic, multi-agent interactions, where safety, performance, and adaptability must be balanced in real time. In overtaking scenarios, an ego vehicle must not only follow a track efficiently but also anticipate and react to the behavior of opponents, whose actions may be suboptimal or inconsistent.

The motivation of this work is to develop an integrated framework that combines robust control with opponent-aware decision-making to enable safe and competitive autonomous racing. To this end, our objectives are to: (i) design a controller capable of stable and safe autonomous driving on a closed track, (ii) enable multi-vehicle interaction through collision avoidance mechanisms, (iii) incorporate opponent behavior prediction based on bounded rationality to support informed overtaking decisions, (iv) develop a high-fidelity simulation environment to validate these methods without real-world risk, and (v) transfer and evaluate the controller in real-world experiments using the F1TENTH car.

C. Problem Statement

In high-speed racing scenarios, being able to predict the behaviors of opponents is crucial both for the safety of your own vehicle and performing high-level behaviors including overtaking or defending your current position. In these situations, opponents are handing multiple objectives and typically do not have enough time or resources to calculate optimal behaviors, leading to these agents picking actions that are "good enough" or "satisfactory". When predicting the behaviors of opponents however many modern approaches assume agents are picking actions that strictly maximize some utility, leading to unrealistic predictions for actions. Therefore, the motivation for this framework is to move towards predicting more realistic behaviors from opponents in high-speed racing for better high-level behavior handling.

II. Literature Review

In this section, we will explore some different relevant literature and explain how it connects to our current work.

A. Information-Theoretic Bounded Rationality in Robotics

There are multiple ways in which bounded rationality can be modeled, and one popular way is through the use of information theory. Under this framework, an agents behavior/strategy for completing a task is represented as a stochastic distribution over possible actions. At any given time, an agent has some initial/instinctual behavior, called the prior policy $p(a)$, and based on its current world state "transforms" their prior policy into an informed posterior policy $p(a|s)$. Following the formulations laid out in [1], decision making is framed as the following optimization problem:

$$p^*(a|s) = \underset{p(a|s)}{\arg\max} \mathbb{E}\big[U(s,a)\big] - \frac{1}{\beta} D_{KL}(p(a|s)||p(a)) \quad (1)$$

$$p^*(a|s) = \frac{1}{Z(s)} p(a) e^{\beta U(s,a)} \quad (2)$$

where β represents the rationality for a given agent, $D_{KL}(p(a|s)||p(a))$ represents the cost from transforming a prior into a posterior, and $Z(s) = \sum_a p(a)e^{\beta U(s,a)}$. This framework has been used in multi-robot teams [2] to model the behaviors of robots with different rationality levels. This approach fell under the assumption that each agent knew the rationality of their teammates and was able to accurately calculate their updated posterior, however in many real world scenarios agents will not strictly know this information. On top of this, rationality in human agents may change over time with one example being a human becoming less rational in their decision making process as they become more tired. Our framework hopes to address these problems by learning the rationality of specific agents online during execution to more realistically predict their behaviors.

B. Model Predictive Control in Autonomous Driving

Model Predictive Control (MPC) is a widely used framework in robotics and autonomous driving, where control inputs are computed by repeatedly solving a finite-horizon optimization problem based on a system model and current state [6]. MPC naturally handles system dynamics and constraints, making it suitable for generating feasible control actions. However, standard MPC formulations typically rely on time-indexed reference trajectories and do not explicitly encode progress along a path. As a result, they are limited in racing-like scenarios, where flexible path following, corner cutting, and overtaking are required. Model Predictive Contouring Control (MPCC) addresses these limitations by incorporating path-relative progress and geometric tracking terms directly into the optimization objective.

III. Approaches and Methods

In this section, we will go into detail about the methods applied in our work and how they are connected to one another.

A. Model Predictive Contouring Control

Model Predictive Contouring Control (MPCC) formulates vehicle control in the spatial domain, balancing path-following accuracy and forward progress along a reference path [3] [4]. Unlike time-indexed MPC, MPCC optimizes geometric tracking error and path progress directly, enabling flexible racing-line selection and aggressive cornering. This makes MPCC particularly suitable for autonomous racing scenarios where strict trajectory tracking is suboptimal due to overtaking and corner cutting. Our implementation is adapted from open-source MPCC frameworks [7]. [1]

1) Spatial Formulation: The racing track is represented by a smooth centerline parameterized by arc length θ. The progress variable θ is included as a system state and evolves as

$$\theta_{k+1} = \theta_k + p_k, \quad (3)$$

where p_k denotes the optimized progress rate. MPCC minimizes geometric errors relative to the reference path: the contouring error e_c and the lag error e_l,

$$e_c = -\sin(\psi_r)(x - x_r) + \cos(\psi_r)(y - y_r), \quad (4)$$

$$e_l = \cos(\psi_r)(x - x_r) + \sin(\psi_r)(y - y_r), \quad (5)$$

where (x, y) is the vehicle position and (x_r, y_r, ψ_r) is the reference pose.

[1] https://github.com/nirajbasnet/Nonlinear_MPCC_for_autonomous_racing

2) Finite-Horizon Optimization: At each timestep, MPCC solves a finite-horizon optimization problem

$$\min \sum_{k=0}^{N-1} \left(w_c e_{c,k}^2 + w_l e_{l,k}^2 + w_u \|\mathbf{u}_k\|^2 - \gamma p_k \right), \quad (6)$$

subject to vehicle dynamics and input constraints. This formulation allows the controller to trade off path accuracy and progress, producing racing behaviors such as smooth cornering and adaptive speed control without a predefined timing reference.

B. Obstacle Avoidance via Soft Constraints

Obstacle avoidance is incorporated directly into the MPCC objective using soft constraints, enabling safe yet flexible navigation in multi-vehicle racing scenarios [4]. Obstacles, including other vehicles, are approximated as geometric regions with safety margins. For an obstacle centered at (x_o, y_o), the violation term is defined as

$$v_o = \max(0, r_o - d_o), \quad (7)$$

where d_o is the relative distance and r_o is the safety radius. The total cost is augmented as

$$J = J_{\text{MPCC}} + \sum_{k=0}^{N-1} \sum_{o} w_o v_{o,k}^2, \quad (8)$$

where w_o controls the aggressiveness of avoidance. This soft formulation preserves feasibility in dense racing scenarios while encouraging collision-free trajectories.

C. Behavior Prediction & Rationality Learning

In order to accurately predict realistic behavior that opponents will take, we need to understand how rational a specific agent is at any given moment. Being able to predict the rationality of opponents will give the ego car a sense of how much effort is being used to strictly maximize the given utility function. When it comes to human agents, this will allow the ego car to adapt to drivers as their rationality changes over the course of a race, with examples including increased tiredness for individuals and switching drivers. In order to implement this behavior, we split this section into two different components, a prediction component and a learning component. The prediction component will compute a prediction trajectory for an opponent based on a current estimate of rationality while the learning component will update the rationality estimate based on observed actions. Information about each respective component can be found below.

1) Prediction Component: While in the previous section the intuition for what information-theoretic bounded rationality, it does not lay out a clear path for implementing this functionality into robotics domains, especially when either the state or action space is continuous. To overcome this hurdle, our method samples potential trajectories using noise injection techniques and than uses importance sampling in order to estimate the posterior distribution and calculate a final predicted trajectory. The nominal control in this technique is equivalent to a default/prior behavior and represents the instinctual behavior an agent will take at any given time. With traditional noise injection methods, random noise sampled from a distribution would be added to every control sequence on a trajectory to generate a set of unique trajectories. This approach, however, would generate unrealistic and dynamically infeasible trajectories, therefore our approach only applies noise a handful of times and execute the previous action. Once a set of trajectories is sampled, each trajectory t is weighed and assign a probability:

$$w(t) = e^{\beta_T U(t)} \quad (9)$$

$$p(t) = \frac{1}{Z(w)} w(t) \quad (10)$$

where β_T represents the current estimate of rationality at time T, $U(t)$ represents the utility of the trajectory, and $Z(w) = \sum_t w(t)$. These trajectories are used to calculate a terminal trajectory that is used as the behavior estimate for an agent and is calculated as follows:

$$u^* = \sum_t p(t) u_t \quad (11)$$

where u_t represents the control for trajectory t. This terminal trajectory represents a prediction for the agent multiple states into the future. In situations where a longer trajectory is needed, this process can be repeated multiple times. Once an predicted trajectory is calculated, the sampling process can be repeated at the end of the current predicted trajectory to extend the prediction further into the future.

2) Learning Component: To ensure that behavior predictions stay accurate, the learning component will observe agents actual actions and update the rationality estimate over time. Once an agent takes an action, the observed action is extended into a trajectory the same length as the sampled trajectories from the previous timestep. A function is used to calculate the closest sampled trajectory t_c to the observed action trajectory, in our case this was based on the distance between state points for trajectories. Bases on this trajectory, the estimate for β_T was updated as follows:

$$\nabla ln(p(t_c; \beta_T)) = U(t_c) - \frac{1}{Z(w)} \sum_t e^{\beta_T U(t)} U(t) \quad (12)$$

$$\beta_{T+1} = \beta_T + \alpha \nabla ln(p(t_c; \beta_T)) \quad (13)$$

The natural log operator is used to handle large differences in probabilities and to simplify certain calculations.

D. Simulation Setup and Environments

To validate the proposed control architecture without risking hardware damage, we developed a customized high-fidelity simulation environment extended from the open-source F1TENTH Gym ROS platform [6]. While existing simulators provide essential sensor emulation (LiDAR and odometry), our implementation introduces several key modifications to support the development of Model Predictive Contour Control (MPCC).

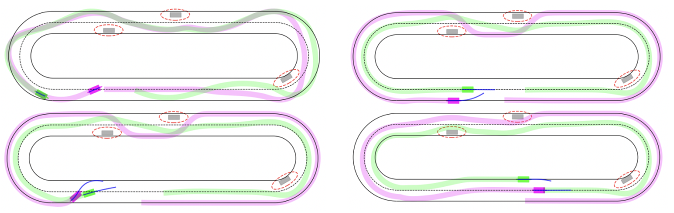
Fig. 2. Effect of MPCC parameter tuning on driving behavior in a low-fidelity racing simulator. Top row: impact of prediction horizon length ($N = 5$ left, $N = 15$ right) with $w_c = 1000$, $w_l = 700$. Bottom row: effect of cost weight tuning, including reduced contouring weight and increased lag weight ($w_c = 500$, $w_l = 1500$, left) and increased obstacle avoidance weight for other vehicles ($w_{\text{other}} = 10^5$, right).

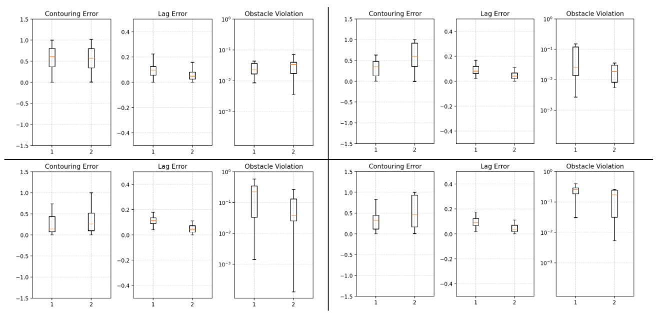

Fig. 3. Distribution of contouring error, lag error, and obstacle violation under different MPCC parameter configurations. Boxplots show error distributions for two vehicles (x-axis labels 1 and 2) simulated concurrently in the same environment. Top row: effect of prediction horizon length ($N = 5$ left, $N = 25$ right) with fixed cost weights ($w_c = 1000$, $w_l = 700$). Bottom row: effect of cost weight trade-offs, comparing contouring-dominant ($w_c = 1000$, $w_l = 700$, left) and lag-dominant ($w_c = 500$, $w_l = 1500$, right) settings.

1) ROS2 Software Architecture for Simulator: This digital twin leverages the real-time modular architecture of the Robot Operating System 2 (ROS2) [8] to replicate the software stack of the actual vehicle. The autonomous driving system is implemented as an independent node that interacts with the simulator through standardized messaging interfaces. The autonomous driving node subscribes to odometry data (/ego_racecar/odom) for state estimation and publishes a set of velocity and steering angle from MPCC to commands (/drive) at a 60Hz rate. Additionally, a map server is integrated to generate a simple path of map through track data, ensuring the simulator provides the accurate spatial reference path needed for the controller's latency error calculations.

2) Real-World Experiment: After testing with simulations, the proposed MPCC framework was applied to a real F1TENTH vehicle to verify its performance on a custom indoor track. During these experiments, a Vicon motion capture system was used instead of onboard SLAM for accurate real-time state estimation. Reflective markers were attached to the vehicle chassis, allowing the Vicon server to track the vehicle's global position. This data was integrated into the ROS2 stack using a bridge node that receives the Vicon stream and converts the coordinates to the vehicle's frame, publishing them as standard odometry (/vicon/car_1/car_1/pose)at an 100 Hz rate. This setup enabled a seamless "simulation-to-real-world" transition. The same autonomous driving node used in the simulation was deployed on the hardware, requiring only a change in topic mapping to subscribe to the odometry data obtained from Vicon instead of the state estimation from the simulator.

IV. Results and Analysis

A. *Effect of MPCC Parameter Tuning*

The effect of MPCC parameter tuning was evaluated in a low-fidelity racing simulator using both qualitative trajectory visualization and quantitative error analysis. Fig. 2 illustrates how changes in prediction horizon length and cost weights affect the resulting driving behavior.

Short prediction horizons ($N = 5$) produce reactive trajectories with limited anticipation of upcoming track curvature, while increasing the horizon to $N = 15$ yields smoother and more anticipatory motion by accounting for future path geometry. In contrast, overly long horizons (not shown) were observed to introduce oscillatory behavior, indicating reduced stability. Cost weight tuning further influences driving style: reducing the contouring weight and increasing the lag weight ($w_c = 500$, $w_l = 1500$) encourages more aggressive progress at the expense of larger lateral deviations, whereas higher obstacle avoidance weights ($w_{\text{other}} = 10^5$) result in conservative trajectories with increased safety margins during interactions.

Quantitative error distributions corresponding to these behaviors are shown in Fig. 3. Increasing the prediction horizon from $N = 5$ to $N = 25$ reduces the spread of lag error, indicating more consistent progress, but increases variability in obstacle violation, suggesting reduced robustness in constrained regions. Cost weight trade-offs reveal a clear balance between contouring and lag errors: contouring-dominant settings reduce lateral deviation but increase lag error, while progress-oriented settings achieve lower lag error at the cost of larger contouring deviations. Across all configurations, obstacle violation distributions reflect the aggressiveness of the tuning, with lower avoidance weights leading to stronger interactions near obstacles.

Overall, these results demonstrate that MPCC parameter tuning directly shapes the controller's driving behavior, even though the underlying control framework remains unchanged. Different parameter priorities induce distinct driving styles ranging from conservative to aggressive, highlighting that controller tuning reflects value-based design choices rather than purely technical considerations. In practice, selecting appropriate parameters requires explicitly defining which objectives—such as safety, smoothness, or competitiveness—should dominate the vehicle's behavior.

B. *Bounded Rationality*

The effect of the rationality coefficient on the overall prediction trajectory was evaluated. Some unique behaviors arise due to the structure of the Gibbs distribution in the original framework and the action-sampling methods used to generate random trajectories. In the discrete variation of information-

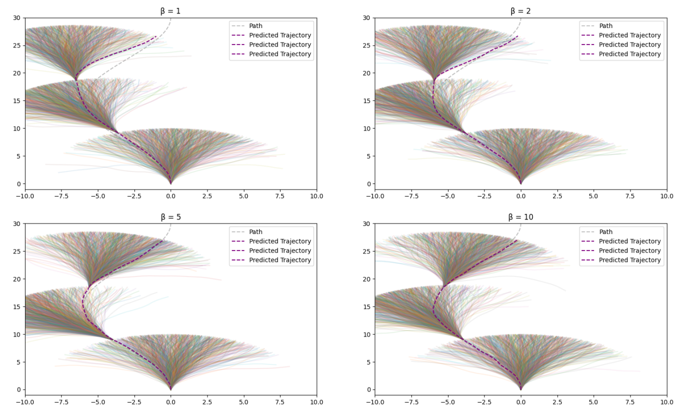

Fig. 4. This figure shows the effect the rationality coefficient β has on the predicted trajectory. When β is 0, the output trajectory follows closely in line with the default/prior behavior, in this case the nominal control is to drive forward. As β starts to increase, the prediction trajectory starts following more closely with maximizing the utility function, in this case following the path.

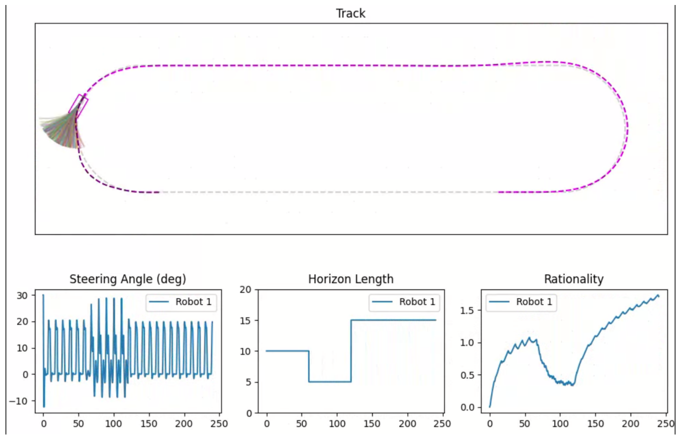

Fig. 5. This figure shows how the predicted rationality for an agent is correlated to the horizon length for a MPCC controller. The utility function used in this experiment was high when on the track and lower as the vehicle moves farther away.

theoretic bounded rationality, setting $\beta = 0$ would collapse the exponential component leaving $p(a|s) = p(a)$ resulting in an agent that does not consider its environment when making decisions and relying on its default behavior. In the case of our sampling based approach, this is equivalent to $w(t) = 1$ for all trajectories, regardless of the calculated utility. This results in an overall prediction trajectory calculated from the average unweighted sum of all sampled trajectories. Since a sampled trajectory is calculated as a nominal term plus a noise term, with the noise term typically being a mirrored distribution around the mean, the default behavior in this case is equivalent to the nominal trajectory unaffected by noise. As the rationality coefficient β starts to increase, trajectories with higher utilities are given more weight when calculating the overall prediction trajectory, leading to this trajectory falling more in line with maximizing the utility function. This behavior can be seen in Fig 4. When $\beta = 0$ the behavior follows the nominal trajectory, in this case a trajectory where the agent does not steer. Once $\beta > 0$, the predicted trajectory falls more in line with the utility function, therefore falling closer in line with following the defined path.

In order to explore learning the rationality for different agents, we explored how the β_T term converged to different terms based on the horizon length for the MPCC controller. A longer horizon length for the MPCC controller allow the controller to consider more of the track at any give time and make plans to follow it around sharp changes and turns while a shorter horizon length only considers a few time steps into the future and only optimizes its short term gain while potentially sacrificing overall gain. Therefore, a short MPCC horizon length should correlate with a low rationality and a longer MPCC horizon length should correlate with a higher rationality. The results for this experiment can be seen in Fig 5. Different horizon lengths for the MPCC controller result in the rationality term converging to different terms. It is also important to note that the ratio between the terms that the rationality coefficient converges to are proportional to the MPCC horizon lengths showing that these two terms are correlated along with our module being able to account for these changes over time. While the ratios are proportional, the value that the rationality coefficient converges to is affected by a variety of factors including the utility function used, the number of trajectories originally sampled, the distribution used to generate noise, the nominal trajectory used, etc. Both in the original framework and our framework, rationality by itself is an abstract term that only has meaning in the context of the domain it is used in and the parameters used. Another important fact to mention is the utility function for weighing trajectories was not the exact same function used for the cost function in the MPCC controller, however this correlation was still present. This means that while following the assumption that agents know the exact utility function opponents are using leads to smoother results, using admissible heuristics can also lead to satisfactory results allowing this framework to be more applicable on real-world systems instead of just simulation-based environments.

C. *High-Fidelity Simulator*

The high-fidelity simulation environment proved to be an essential tool in the development of the MPCC framework. Acting as a bridge to the physical system, the simulator allowed us to adjust and test key cost function weights, particularly the contouring error e_c and the lag error e_l. As shown in Fig 6, this iterative adjustment process successfully yielded a stable set of parameters that maximized theoretical driving performance while adhering to the vehicle's kinematic constraints, providing a validated baseline for real-world system implementation.

D. *Real-World Experiment*

In real-world experiments, the system demonstrated robust "simulation-to-real-world transfer" capabilities. By combining a modular ROS 2 architecture with a high-precision Vicon state

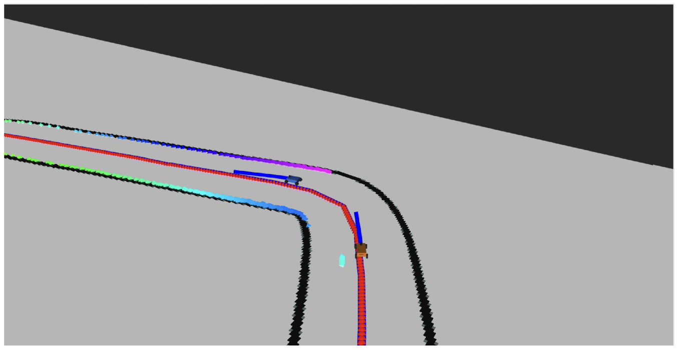

Fig. 6. This figure shows how the MPCC controller worked in the high-fidelity simulator. The blue line shows the horizon length of the MPCC controller and the horizon length tries to follow the red path.

estimation system, the control logic developed in simulation could be directly applied to the F1TENTH vehicle. Fig 1 shows the real vehicle successfully navigating a custom track, confirming that the aggressive cornering behavior and stability margins observed in simulation are reproducible on the actual hardware. This demonstrates that the proposed approach provides a computationally feasible solution for autonomous racing on embedded platforms.

V. Conclusion and Future Work

Throughout our project, we developed a framework to start moving closer to more realistic behavior predictions in autonomous racing scenarios. We incorporated an MPCC controller to follow a designated path around a race track and explored the impacts of different tuning parameters to the overall performance. We incorporated a behavior prediction module using information-theoretic bounded rationality that consists of two separate components, a prediction component to generate prediction trajectories based on an estimated rationality, and a learning component to update the rationality prediction over time. Finally, we tested our methods and results in three different environments: a low-fidelity simulation environment for quick experimentation, a high-fidelity simulation environment implemented in ROS2, and a real-world environment using F1Tenth robots and a Vicon tracking system.

VI. Contributions

Tristan Scheiner worked on initially building and setting up the low-fidelity environment. This was implemented in python using matplotlib and included both a track and data graphs for visualization. The purpose of this was to have a simulation environment to quickly run experiments and test different methods. He also designed and implemented the behavior prediction section using information-theoretic bounded rationality. This consisted of two different components, a prediction component and a learning component, and the motivation was to start moving towards more realistic behavior prediction methods in high-speed racing. Finally, he helped setup and run experiments on the physical F1Tenth car using ROS2. One potential area for future work includes enhancing the MPCC optimization process using the predicted trajectory from the bounded rationality module. Another area includes learning the tuning weights for the system through a combination of imitation and reinforcement learning.

Jiyeong Oh contributed to the development and refinement of the MPCC-based control framework, including the integration of collision avoidance via soft constraints. She improved the low-fidelity racing simulator to support multi-vehicle interactions, obstacle modeling, and detailed logging for analysis. Jiyeong conducted extensive parameter tuning experiments to analyze the effects of horizon length and cost weight trade-offs on driving behavior, and led the quantitative and qualitative evaluation presented in this report. In addition, she assisted with real-world experimental setup and data collection to support the validation of the proposed approach.

Yongjun Cho worked on exploring high-fidelity simulators such as AutoDrive, but our team couldn't use the simulator due to lack of documentation and physical limitation of local machine. Therefore, he worked on the current high-fidelity simulator, making codes with ROS2 and tuning parameters from the MPCC controller to make the controller work well in the simulation. In addition, he helped with the real-world experimental setup and experiments on the physical F1Tenth car using ROS2.

References

[1] T. Genewein, F. Leibfried, J. Grau-Moya, D. Braun, *Bounded Rationality, Abstraction, and Hierarchical Decision-Making: An Information-Theoretic Optimality Principle*, Front. Robot. AI, 2015

[2] J. Xu, D. Pushp, K. Yin, L. Liu, *Decision-Making Among Bounded Rational Agents*, Distributed Autonomous Robotic Systems, 2024

[3] D. Lam, C. Manzie, and M. Good, *Model Predictive Contouring Control*, in Proc. 49th IEEE Conf. on Decision and Control (CDC), 2010, pp. 6137–6142.

[4] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, *Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments*, IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 4459–4466, 2019.

[5] Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. Automatica, 36(6), 789-814.

[6] F1TENTH Foundation, *f1tenth_gym_ros: ROS2 wrapper for F1TENTH Gym*, https://github.com/f1tenth/f1tenth_gym_ros.

[7] N. Basnet, *Nonlinear MPCC for Autonomous Racing*, https://github.com/nirajbasnet/Nonlinear_MPCC_for_autonomous_racing

[8] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, *Robot Operating System 2: Design, architecture, and uses in the wild*, Science Robotics, vol. 7, no. 66, 2022.