A critical evaluation of Pure Pursuit, MPC and MPCC: balancing simplicity, performance and constraints

Xiangyiming Kong[1*], *Callen* Fisher[1], and *Benjamin* Evans[1]

[1]Stellenbosch University, Electrical and Electronic Engineering Department, Western Cape, South Africa

Abstract. Autonomous driving technology has advanced significantly in the past decade. In the case of autonomous racing, the crucial challenge lies in guiding the vehicle along the desired path while ensuring safety and efficiency. Several control systems have been developed for fast and accurate path tracking, offering their own unique advantages and limitations. Here we present an investigation into three prominent control strategies in a critical comparative analysis. These control algorithms include Pure Pursuit (PP), Model Predictive Control (MPC) and Model Predictive Contouring Control (MPCC). These control algorithms were chosen due to their difference in complexities starting from a simple PP to a much more complex MPCC. These algorithms are then experimentally validated on a physical F1Tenth vehicle. The simulation results show that PP exhibited the fastest computation time, its performance in the presence of noise and delay was inferior to MPC and MPCC. While MPC demonstrated strong performance in its robustness and resilience to noise and delay compared to PP and MPCC. MPCC, despite producing the fastest lap time, faced challenges in handling noise and delay although not as severe as PP, making MPC the best overall controller for unwanted disturbances. In contrast to the simulated findings, PP demonstrated superior performance in physical implementations compared to MPC and MPCC. The computational demands of optimization-based algorithms result in a computation delay where the algorithm calculates for the vehicle's position one iteration behind its actual movement. This shows that shortening computational delay is critical in a physical system.

1 Introduction

With advancements in technology, autonomous vehicles are becoming increasingly capable of navigating complex environments without human intervention. In an autonomous vehicle, the control pipeline can be separated into three categories, namely: perception, planning and control [5]. The pursuit of optimal control methods in autonomous racing has led to the development of various approaches, each offering unique advantages and limitations. Pure Pursuit (PP) [1], Model Predictive Control (MPC) [2] and Model

* Corresponding author: 22902716@sun.ac.za

Predictive Contouring Control (MPCC) [3] represent the three prominent methods in this domain [4]. Within these three algorithms, PP and MPC are control strategies which follows a pre-determined trajectory, while MPCC does the path planning and actuation control online. The difference between the PP and MPC is that the PP control algorithm focuses on steering the vehicle towards a desired point in a global planner based on the lookahead distance. While MPC uses its predictive modelling nature to anticipate the future states and optimize a control strategy accordingly. MPCC as the name suggests, takes a concept of contouring control into consideration which includes the optimization of the progress along the center line while minimizing the cross track error.

This research follows the F1Tenth framework [6], an LiDAR based perception system, which uses Adaptive Monte Carlo localization [12] and Hector Simultaneous Localization and Mapping (SLAM) [13] algorithms to localize and map out the surrounding area. Then, using a minimum curvature optimization algorithm, a race spline was calculated. This spline contains the Cartesian coordinate for the racing trajectory and the speed profile, which is used as the tracking line for PP and MPC controllers.

In the simulation phase, the parameters were tuned to the specific track where the optimal parameter was defined as the parameter that gave the best lap time and tracking accuracy. This was used as a benchmark for comparing the effects of the system noise as well as perception and computational delays. The same parameters were then run on the physical F1Tenth vehicle to study the difference in performance between simulation and the physical vehicle. Then, a critical analysis was done based on the results to validate which control algorithm was the most robust and reliable on the physical car.

2 Related work

The recent surge in the development of electric vehicles has propelled significant interest in autonomous vehicles. Researchers are actively exploring various methods to ensure the safe and reliable control of these vehicles.

In [7] and [8], researchers have explored the application of PP control algorithms, employing both simulation and real-world testing environments. Notably, [8] introduced an innovative approach by assigning an optimized look-ahead distance to each waypoint in the planning stage. The findings show promising outcomes, showcasing the efficacy of the adaptive look-ahead PP control algorithm in reducing lap times and enhancing average lap speeds compared to normal PP implementation. Renault Zoe electric vehicle was used in [7] as the experimental vehicle, the physical results showed the average straight-line error was 0.15 m and 0.63 m at the shallow corner. This error is acceptable for road use as the vehicles are not travelling very fast. The work in this paper further investigates the trajectory and speed profile following error under high-speed conditions and utilizes an online adaptive lookahead method inspired by [8].

In [15], a Genetic Algorithm based Nonlinear MPC (NMPC) system was deployed, showing promising results when applied to road vehicles. The evaluation reveals that the NMPC algorithm yields a tracking error with a standard deviation of 0.13 m overall. In comparison with [7], NMPC has a better tracking error around corners than the PP control method. An improved MPC was implemented in [16] and [17]. [16] uses a fuzzy logic dynamic weigh adjustment in the cost function to create a smoother trajectory and [17] uses a softened constraint system to improve the solution finding process. Both methods improve the smoothness of the steering and acceleration for maximum driver comfort and quick convergence to the tracking trajectory.

Researchers have also proposed several improved strategies for MPCC control. Similar to [16], [19] also uses fuzzy logic to dynamically change the weighting in the cost function to improve the trajectory tracking line. [20] on the other hand uses a different approach to adjust the weighting. This method uses a Reinforcement Learning agent as the observer to process the perception data. Based on the car's position, the agent adjusts the parameters to achieve the best possible trajectory and tracking control.

In this research, a barebone version of each control algorithm was implemented and a comparative analysis was made to investigate each of their advantages and limitations. Hoping to find which control algorithm is the best suited for different scenarios.

3 Model formulation

In this section, the model formulation of the F1Tenth vehicle is presented [10]. It aims to describe the simulation model and the physical vehicle's hardware component. The simulation model is based on the bicycle kinematic model, which is used to present a simplified model of the F1Tenth RC vehicle. This simplified model allows the researchers to develop and test control algorithms without the need of working with complex nonlinear systems that exist in the real world. Following this, the physical hardware of the F1Tenth vehicle is discussed. Then, the formulation of the trajectory planning algorithm, Minimum Curvature trajectory, is explained as this is used as the primary path planning algorithm for this study.

3.1 Simulation model formulation

In reality, the dynamics of a high-speed vehicle are nonlinear and difficult to model. Due to inaccurate weight alignment, component noise and dynamic friction coefficients on uneven surfaces, the physical vehicle may behave in unexpected ways. With these complications, accurate modelling of the physical vehicle can cause significant computational costs, resulting in prolonged computation time and expensive hardware. Therefore, a simplified linear model was needed for fast computation while maintaining similar dynamics of the physical vehicle. For this purpose, the bicycle kinematic model was chosen as the vehicle dynamic model [9].

The bicycle kinematic model is widely used as the representation of vehicle dynamics that forms the basis of the simulation model for the F1Tenth vehicle [10]. The model takes a 4-wheel vehicle model and combines the two front wheels and two rear wheels to form a 2-wheel model shown in Fig. 1. In order to make this simplification possible, some important assumptions must be made. The vehicle only operates in a 2D plane, the vehicle has lumped all its mass at the center, and it is assumed that there is no lateral or longitudinal slip.

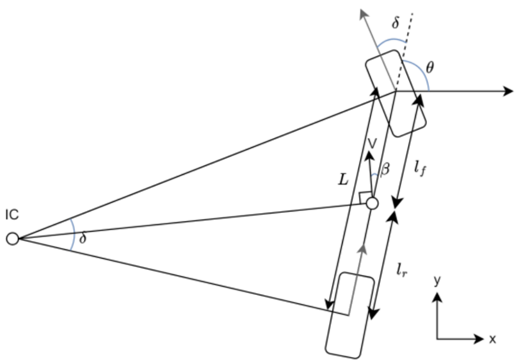

Fig. 1. Schematic of the kinematic bicycle model.

The state dynamic equations for the kinematic model in Fig 1 is as follows:

$$\dot{x} = v * \cos(\theta + \beta), \tag{1a}$$
$$\dot{y} = v * \sin(\theta + \beta), \tag{1b}$$
$$\dot{\theta} = \omega = v * \frac{\tan\delta \cos\beta}{L}, \tag{1c}$$
$$\beta = \arctan\left(\frac{l_r \tan\delta}{L}\right). \tag{1d}$$

In Eq.1, $\dot{x}$ and $\dot{y}$ are the linear velocity of the vehicle on the x and y axis, θ is the heading angle of the vehicle and v is the lateral velocity of the vehicle. L denotes the wheelbase, the distance between the front and the rear axles, and δ is the steering angle of the front wheel.

The bicycle kinematic model uses two inputs to calculate its next state. These are velocity of the vehicle v and steering angle δ. For this purpose, the model is discretized, using information of the current state to predict the next state, where Δt denotes the sample time of the system. The update equations used are as follows:

$$x(t+1) = x(t) + \dot{x}(t) * \Delta t, \tag{2a}$$
$$y(t+1) = y(t) + \dot{y}(t) * \Delta t, \tag{2b}$$
$$\theta(t+1) = \theta(t) + \dot{\theta}(t) * \Delta t, \tag{2c}$$
$$\delta(t+1) = \delta(t) + \dot{\delta}(t) * \Delta t. \tag{2d}$$

3.2 Physical model setup: F1Tenth

To realize the simulated model, the F1Tenth RC vehicle was chosen [6]. F1Tenth is an open-source project which provides the physical and software stack to create a one-tenth scale of an autonomous vehicle. The autonomous car is designed specifically for racing which fits the objective of this study, as the objective involves pushing the vehicle to its limits and study its behavior. The physical setup of a F1Tenth vehicle is designed to be robust, agile, and capable of high-speed maneuvers.

The physical layout of a F1Tenth vehicle typically includes:

- Traxxas Slash 4X4 Chassis: Provides the vehicle's basic structure, including the frame, battery, and the suspension.
- 1/10 Scale Brushless Pro 4WD Motor: The F1Tenth car is powered by a brushless DC motor with an Electronic Speed Controller (ESC) controlling the speed of the motor from the signals sent from the onboard computer.
- Hykuyo 10LX LiDAR: Used for perception, generates a 2D scan of the surrounding. The data from the LiDAR is crucial for localization and obstacle detection.
- Jetson Xavier NX Onboard computer: The vehicles brain which processes the information from the sensors and outputs instructions to the ESC to actuate the vehicle.

3.3 Center line and minimum curvature trajectory planning

In the planning stage of the autonomous pipeline, a minimum curvature trajectory planning algorithm adapted from [21] is used in this research. Using the LiDAR data gathered, the mapped area from the track is modified into an occupancy grid map. Each grid of this image is classified using Boolean logic. This grid image is then smoothed and filtered to try find the center line's true x, y coordinates and its corresponding track width [21].

The derivation of the race line is split into two problems, the minimum curvature path is first calculated, then the speed profile is calculated given the curvature of the optimized path. To calculate the minimum curvature path, the cost function 3a is formulated:

$$J = \sum_{n=1}^{N} \gamma_i^2(t), \tag{3a}$$

$$x_i(t) = a_i + b_i t + c_i t^2 + d_i t^3, \tag{3b}$$

$$y_i(t) = a_i + b_i t + c_i t^2 + d_i t^3, \tag{3c}$$

$$\gamma_i = \frac{x_i' y_i'' - y_i' x_i''}{(x_i'^2 + y_i'^2)^{\frac{3}{2}}}. \tag{3d}$$

Where the curvature of the path (γ_i) is the objective of the optimization problem. To find the minimum curvature path for the racetrack, a quadratic optimization problem can be formulated. The race line is defined in the quadratic problem using a third order spline of its x and y coordinates shown in 3b and 3c respectively. Using these coordinates, the first order and second order derivative are derived with respect to time and substituted into the discrete evaluation point equation, 3d, to calculate the curvature of the race line. Since each consecutive point of the center line are required to be connected, their first and second derivative must also be continuous. Using these quadratic equations, the spline parameters a_i, b_i, c_i and d_i can be calculated using the solutions of the linear equations.

Using the derived race line, the velocity profile can now be calculated. This uses the acceleration envelope approach which considers longitudinal and lateral acceleration limits of the car at different speeds. This velocity profile must also consider the tire friction and slip conditions. Therefore, a forward-backward-solver is implemented for this optimization. This solver calculates two velocity profiles, a forward and backward solution which are then intersected. Initially, the solver estimates the velocity profile based on the curvature profile of the race line. Then, the speed profile is cut off at the maximum speed allowed for the vehicle. Once a good estimate of the velocity profile has been formulated, the forward calculation procedure can be started. This procedure modifies the velocity profile to keep positive longitudinal and lateral acceleration within the car's limit. The backward calculation will do the same but for negative longitudinal and lateral acceleration limits.

4 Control methods

4.1 Pure Pursuit

Pure Pursuit is a path tracking algorithm that is widely used in autonomous vehicles. The vehicle is guided along a global path designed by a planner and constantly chasing after a "pursuit waypoint" at a given lookahead distance. The algorithm calculates the steering angle based on its current position, the orientation of the vehicle and the location of the pursuit point shown in Fig 2.

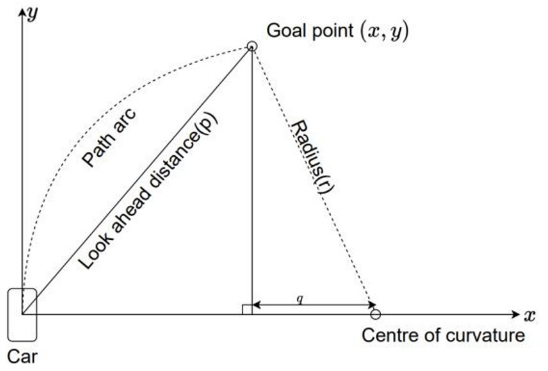

Fig. 2. Pure Pursuit steering calculation diagram.

$$\delta = \arctan\left(\frac{2xL}{p^2}\right). \tag{4}$$

Since the speed of the vehicle is predetermined by the global planner, using the lookahead distance *(p)* and the vehicle length (L), steering angle (δ) can be calculated.

4.2 Model Predictive Control

Model Predictive Control is a widely used control method in robotics, it is an optimal control technique which calculates the control actions by minimising a cost function for a constrained dynamic system over a finite time horizon. At every time step, the MPC controller will receive the current position and calculates the sequence of control actions that minimizes the cost function over the time horizon, N, by solving a constrained optimization problem. Once the optimiser calculates the solution, the first control action is sent as the actuation command, while the optimisation process repeats.

The MPC uses the bicycle kinematic model, the input and output of the system is shown as follows:

$$U_{k+1} = f(x, y, \phi, v), \tag{5a}$$
$$y_k = g(\delta, a). \tag{5b}$$

Where U_{k+1} denotes the system's next observable state which is dependent on the current pose of the vehicle (x, y), yaw angle (ϕ) and the current speed (v). The actuation command y_k for the current state, which consist of steering angle (δ) and acceleration (a), this is calculated using the IPOPT [11] solver.

The cost function is set up to minimize the tracking error of the trajectory and the speed profile,

$$J = \sum_{n=1}^{N}\big(f(x, y, \phi) - ref(x, y, \phi)\big)^2 + \big(f(v) - ref(v)\big)^2, \tag{6a}$$
$$\delta_n \in [-0.4, 0.4]rad, \tag{6b}$$
$$v_{n+1} \in [0,8]m/s, \tag{6c}$$
$$n = 1, 2, 3, \dots, N. \tag{6d}$$

The trajectory tracking error is defined as the difference between the vehicles current position to the reference trajectory ($ref(x, y, \phi)$), and speed profile error is the difference between the current speed and the reference speed profile $\big(ref(v)\big)$.

4.3 Model Predictive Contouring Control

Model Predictive Contouring Control (MPCC) is a more sophisticated algorithm for achieving precise and robust motion control in a complex nonlinear dynamical system. Like MPC, MPCC operates over a finite prediction horizon, and formulates an optimization problem aiming to minimise the cost function. The key difference between MPCC and MPC is that MPCC is able to bridge the gap between trajectory planning and the feedback control in the control pipeline. While MPC minimises tracking error, MPCC takes a step further. It considers the system's progress along the centre line, a concept known as contouring. While minimizing the tracking error, MPCC also aims to make smooth progress along the overall path.

The cost function set up for MPCC is as follows:

$$J_k = \sum_{i=1}^{N} \hat{\epsilon}_{cont}^2 Q_{cont} + \hat{\epsilon}_{lag}^2 Q_{lag} - Q_\theta \theta_{k+i} + \Delta u_{k+i} Q_u, \quad (7a)$$
$$x_{k+i} = x_{k+i-1} + u_{k+i-1}, \quad (7b)$$
$$\theta_{k+i} = \theta_{k+i-1} + v_{k+i-1}, \quad (7c)$$
$$u_{k+i} \in [-0.4, 0.4]\, rad. \quad (7d)$$
$$v_{k+i-1} \in [0, 8] m/s, \quad (7e)$$
$$\theta_{k+i} \in [\theta^s, 0]. \quad (7f)$$

$\hat{\epsilon}_{cont}$ and $\hat{\epsilon}_{lag}$ denotes the linearized contour error and lag error respectively. Q denotes the weighting parameter for each of the variables. θ_k denotes the progression along the center line, its evolution is governed by $\theta_{k+1} = \theta_k + v_k$. Where θ_k denotes the value of the path parameter at time k and $v_k$denotes the virtual input to be determined by the controller. Lastly, Δu_{k+1} denotes the control effort, minimizing sudden change in steering and velocity. Fig. 3 shows that the contour error and lag error are nonlinear and need to be approximated to make the computations more simplified.

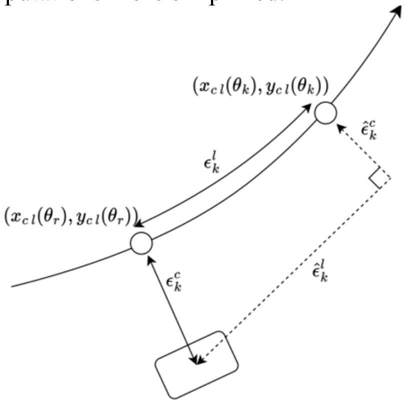
$(x_{cl}(\theta_k), y_{cl}(\theta_k))$
$\hat{\epsilon}_k^c$
ϵ_k^l
$(x_{cl}(\theta_r), y_{cl}(\theta_r))$
$\hat{\epsilon}_k^l$
ϵ_k^c

Fig. 3. Contouring error and lag error schematic.

The linearized contouring error $\hat{\epsilon}_{cont}$ and $\hat{\epsilon}_{lag}$ is therefore approximated by 7a and 7b respectively.

$$\hat{\epsilon}_{cont} = \sin(\phi(\theta_k))\,(x_k - x_{cl}(\theta_k)) - \cos(\phi(\theta_k))(y_k - y_{cl}(\theta_k)) \quad (8a)$$
$$\hat{\epsilon}_{lag} = -\cos(\phi(\theta_k))\,(x_k - x_{cl}(\theta_k)) - \sin(\phi(\theta_k))(y_k - y_{cl}(\theta_k)) \quad (8b)$$

5 Simulation results

In order to evaluate the three control algorithms, their respective parameters were optimized across diverse maps in simulation. Next, the effectiveness under realistic conditions were assessed in simulation by introducing noise and delays into the perception input and actuator output, mimicking the real-world sensor inaccuracies and hardware limitations. This approach evaluates both efficiency and robustness of the three control algorithms.

5.1 Parameter fine-tuning

Since every map differs slightly on the width of the track and the curvature of the corners, the parameters should be slightly adjusted to account for this difference. The optimal parameters were defined by the set of parameters that produced the fastest lap and the lowest average tracking error. This process involved using Monte Carlo experiments, which involved repeatedly running the simulator while randomizing a range of values. These parameters were used as a baseline measurement to compare the effects of noise and delay in the experiments.

Through these experiments a notable remark was observed. The PP control algorithm tended to oscillate out of control if the look-ahead distance was too close and cut corners if the look-ahead distance was too far.

A simple solution was made to prevent oscillation and corner cutting. By making the look-ahead distance (l_d) a function of speed, resulting in the point of pursuit being closer when the vehicle is taking a corner, preventing corner cutting and further away when the vehicle is travelling along a straighter line, preventing oscillation.

$$l_d = l_{gain} v + l_{const} \tag{9}$$

Where l_{gain} and l_{const} are the tuneable parameters for PP.

5.2 Noise and delay

To evaluate the robustness of the car's control system, noise and delay were introduced into the perception data and the control commands, in simulation. These disturbances were aimed to simulate the real-world scenarios where inaccurate sensor measurements are frequently encountered.

The F1Tenth vehicle uses an Adaptive Monte Carlo Localization (AMCL) [12] as its localization method and Hector Simultaneous Localization and Mapping (SLAM) [13] as the mapping method. Due to their probabilistic nature, the final output of the vehicle position is not completely accurate. From [12] and [14], the position error from localization with MCL and the mapping with Hector SLAM combined in an indoor setting can be up to 10 cm which can be significant for the F1Tenth vehicle, therefore Gaussian noise with $\mu = 0$ and $\sigma = 0.2$ were selected as the disturbance.

Inaccuracies in the actuation of the vehicle can also cause sub-optimal performances, since the motors use a PID controller, the actual speed and steering cannot match the reference speed and steering perfectly. The reference commands are quickly changing as the car is racing through the track. Therefore, it is necessary to critically analyse how robust each of the control algorithms are to the inaccuracies cause by the PID controller. Both perception noise and control command noise are summed with the true position given by the simulation environment and the control command determined by the algorithm respectively, to study the effect of noise on the performance of the simulated vehicle.

Additionally, this research examines the effect of delays in perception data as well as the control commands on the F1Tenth car. Delays are introduced to mimic the communication latency that may occur in real-world situations. This includes the pose computation time from AMCL, the computation time for the control algorithm to calculate the optimal action and the actuation time the motor takes to match the reference actuation command sent by the control algorithm. Each disturbance was inserted independently into the control system. For each disturbance inserted, the simulation runs for 20 laps and the average results were recorded. Then, the delay duration is increased at 10ms increments starting from 0ms to 100ms.

5.3 Results

In the section, results were obtained from simulations in the F1Tenth Gym environment, then the comparison between the performance of the PP, MPC and MPCC were analysed. The algorithms were evaluated on three different F1 tracks: Austria, British and Spain Grand Prix tracks. The objective of these experiments is to evaluate the algorithms' resilience to noisy and delayed perception and actuation data. This evaluation uses the track progress, lap time and the deviation from the optimal path on the F1 tracks as the measurement metric.

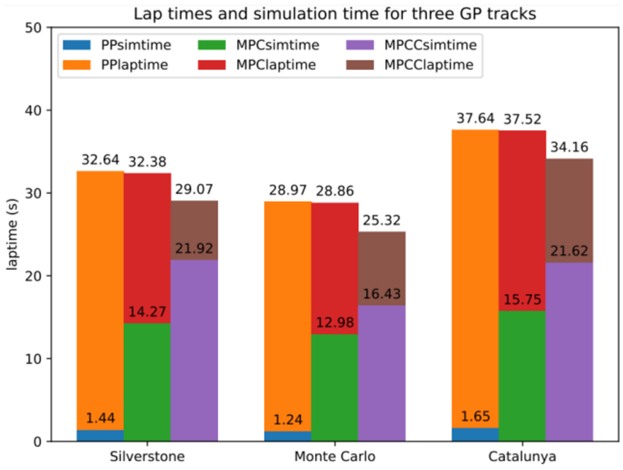

Fig. 4. Lap time and simulation time for PP, MPC, MPCC on 3 different GP tracks.

Fig. 4 shows that PP had the fastest computation time but the slowest lap time. This is because PP is the simplest control algorithm as it only does one iteration of calculation based on its look-ahead distance. MPC and MPCC on the other hand must run through iterations of optimisations to arrive at the optimal solution, thus causing an increase in computation time. Between the two algorithms, MPC produced a slightly slower lap time than PP, but is much faster in computation time compared to MPCC. However, MPCC outperformed both PP and MPC in lap time, on average 3 seconds faster. This is due to MPCC having a much more aggressive speed profile, as shown in Fig. 5, taking a more ideal line than the theoretical geometric line used in PP and MPC. MPCC calculates its speed profile differently to PP and MPC. MPCC calculates its speed online given its current position, the algorithm uses a simple tire model and determines its speed by its current lateral acceleration. Different to the Minimum Curvature trajectory, calculation was done offline and only uses the curvature of the path as a dependent variable. This allows the vehicle to travel at the maximum allowed speed and can dynamically change depending on where the vehicle is relative to the track.

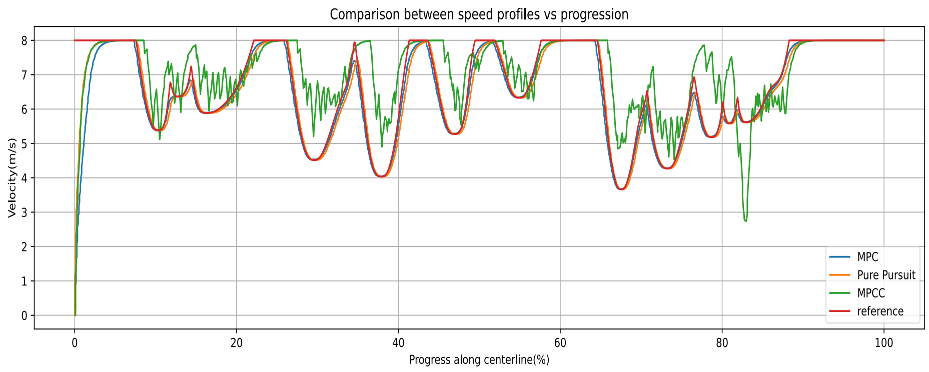

Fig. 5. A typical speed profile for PP, MPC and MPCC algorithm.

Following this, the robustness of each control algorithm is analysed, the Fig. 6 to Fig. 11 shows the average track progress with one disturbance inserted at a time. The performance is determined by the length of the vehicle was able to cover before it crashed against the boundary, further it is able to go the more robust that algorithm is to that specific set of disturbance. In addition to the average track progress, the cross track error is also used as a metric to measure the robustness of a control algorithm to disturbances. However, MPCC does not track any specific path, therefore cross track error is not applicable to MPCC but it is a useful tool to see the effects of disturbance on PP and MPC.

The simulated results show that PP is sensitive to noise and delay, this is evident in Fig. 6, 7, 8, and 9, with limitations in handling perception and steering control noise and delays. While PP displayed robustness in completing laps under speed control noise and delay, it struggled with perception and steering control noise, indication vulnerability in noisy environments.

MPC's ability to handle delays, especially in perception and steering control, outperformed PP shown in Fig. 6, 7, and 9. It demonstrated resilience in completing laps under various noise conditions, highlighting its adaptability to noisy environments. However, MPC's complexity let to longer computation times compared to PP, posing a trade-off between efficiency and performance.

MPCC showed reliability in completing laps under perception noise and steering control noise in Fig. 6 and 8, indicating robustness in noisy condition. However, its complex cost function results in the longest computation times, limiting its efficiency compared to PP and MPC. MPCC faced challenges in handling speed control noise and delays in Fig. 10 and 11, lagging behind the other algorithms in this aspect.

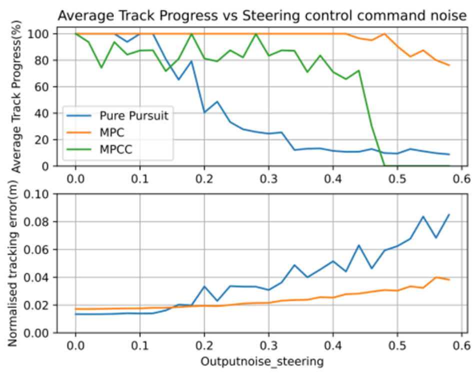

Fig. 6. Average track progress on Catalunya with steering control noise.

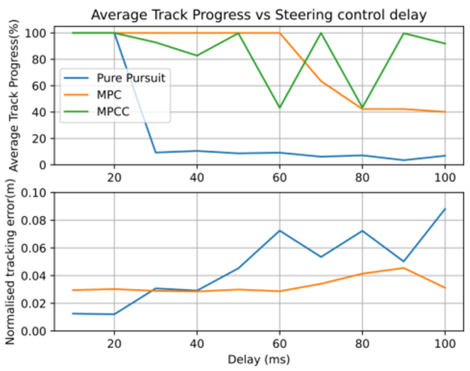

Fig. 7. Average track progress on Silverstone with steering control delay.

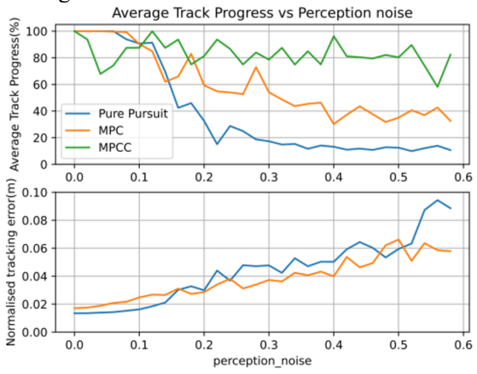

Fig. 8. Average track progress on Catalunya with perception noise.

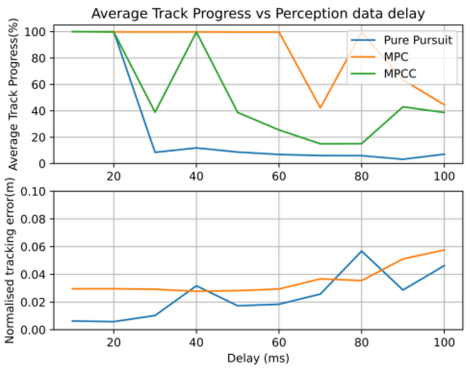

Fig. 9. Average track progression on Silverstone with perception delay.

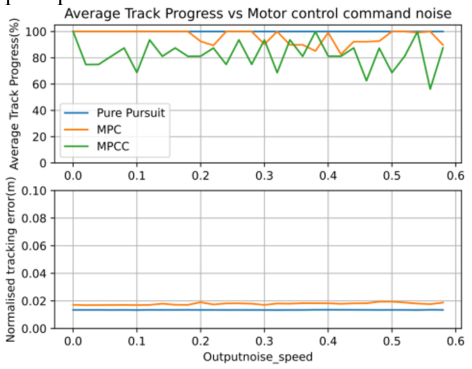

Fig. 10. Average track progress on Catalunya with motor control noise.

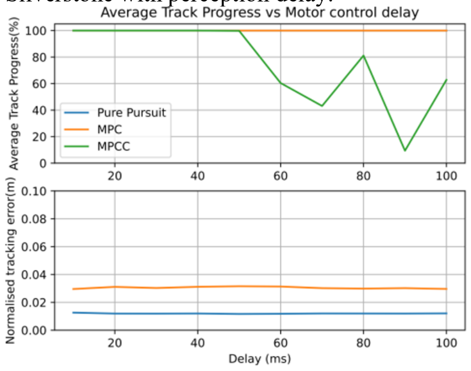

Fig. 11. Average track progress on Silverstone with motor control delay.

In conclusion, after a thorough analysis, MPC has demonstrated strong performance in various scenarios, indicating its robustness and resilience to noise and delay compared to PP and MPCC. While Prue Pursuit exhibited the fastest computation time, its performance in the presence of noise and delay was inferior to MPC and MPCC. MPCC, despite producing the fastest lap time, faced challenges in handling noise and delay, making MPC the most suitable controller overall for the simulated F1Tenth car.

6 Physical test results

In contrast to the simulated results, the physical experimentation revealed a different hierarchy of performance among the control algorithms. The physical results were gathered from the physical F1Tenth RC vehicle as discussed in Section 3.2. As shown in Fig. 12, Pure Pursuit emerged as the top performer, followed by MPCC, then MPC. This unexpected outcome results from the inherent simplicity of the PP algorithm. In the real-world scenario, computation resource is often the constraint. Due to PP's shorter computation time, it can send control commands at a much higher frequency, resulting in more accurate control. On average, PP sends a control command at a frequency of 8.77 Hz, while MPC does so at 5 Hz and MPCC at 3.07 Hz. With a vehicle moving at 5 m/s, in 100 ms the vehicle can cover 0.5 meters. This discrepancy can lead to the algorithm planning for position that occurred 100 ms ago, causing observed oscillations shown in Fig. 12.

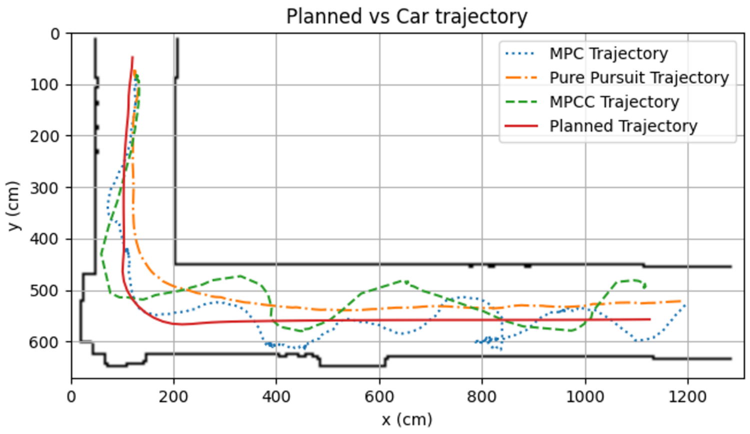

Fig. 12. Physical vehicle trajectories with PP, MPC and MPCC algorithms.

On the other hand, due to the more sophisticated nature of MPC and MPCC, although theoretically advantageous in terms of optimization and trajectory planning, introduced computation complexities that hindered their real-time responsiveness. Despite MPCC demonstrating commendable speed in completing laps in simulation, its performance was surpassed by PP due to the latter's ability to deal with much higher delay than given in simulation. Similarly, MPC, while offering robust control in simulated environments, struggled to translate its theoretical prowess into tangible real-world advantages, largely due to the impact of computation delays.

The disparity in performance highlights the critical role of computation delay in an autonomous vehicle system. These delays resulting from the time taken to process sensor data, compute control commands and execute actions, can significantly affect the overall efficiency and effectiveness of control algorithms. In fast paced environments such as racing circuits, where split-second decisions determine success or failure, minimizing computation delays becomes imperative. While complex algorithms like MPC and MPCC may offer superior theoretical performance, their practical computation, underscoring the importance of striking a balance between algorithmic sophistication and real-time responsiveness in autonomous driving systems.

7 Conclusion

This research paper has demonstrated that Model Predictive Control (MPC) emerges as the most robust controller for the F1tenth framework, outperforming PP and Model Predictive Contouring Control (MPCC) in simulated scenarios. Despite PP's computational efficiency, MPC's resilience to noise and delay makes it the superior choice overall. However, MPC controllers faces difficulty when the command frequency is low, this is where the simplicity of PP algorithm shines. With its simple calculations it is able to control the car at a higher frequency, thus more accurate control.

However, the disparity between simulation and physical testing highlights the need to bridge this gap for more accurate evaluations. Future work will focus on minimizing this difference, enhancing hardware capabilities, and refining optimization methods to improve real-world applicability.

References

1. M. Behl, V. Sukhil, *Adaptive Lookahead Pure-Pursuit for Autonomous Racing,* CoRR, abs, **2111.08873** (2021)
2. T.M. Vu, R. Moezzie, J. Cyrus, J. Hlava, *Model Predictive Control for Autonomous Driving Vehicles*, *Electronics*, ***10***, 2593 (2021)
3. D. Lam, C. Manzie, M. Good, *Model predictive contouring control*, 49th IEEE CDC, 6137-6142 (2010)
4. J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, R. Mangharam, *Autonomous Vehicle on the Edge: A Survey on Autonomous Vehicle Racing,* IEEE Open Journal of Intelligent Transportation Systems, **3**, 458–488 (2022)
5. S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, M. H. Ang, *Perception, Planning, Control, and Coordination for Autonomous Vehicles,* Machines, **5**, 1, 6, 2075-1702 (2017)
6. M. O'Kelly, H. Zheng, D. Karthik, R. Mangharam, *F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning,* PMLR, **123**, 77-89 (2020)
7. D. Kapsalis, O. Sename, V. Milanes, J. Molina, *Design and experimental validation of an lpv pure pursuit automatic steering controller,* IFAC, **54**, 2, 63-68 (2021)
8. M. Behl, V. Sukhil, *Adaptive Lookahead Pure-Pursuit for Autonomous Racing,* CoRR, abs, **2111.08873** (2021)
9. P. Polack, F. Altche, B. d'Andr´ea Novel, A. de La Fortelle, *The Kinematic Bicycle Model: a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles?,* 2017 IEEE IV Symposium, 812-818 (2017)
10. A. Jain, M. O'Kelly, P. Chaudhari, M. Morari, *BayesRace: Learning to race autonomously using prior experience,* CoRR, abs, **2005.04755** (2020)
11. A. Waechter and C. Laird, "Ipopt documentations," (2023). [Online]. Available: https://coin-or.github.io/Ipopt/
12. W. Xiaoyu, L. Caihong, S. Li, Z. Ning, F. Hao, *On Adaptive Monte Carlo Localization Algorithm for the Mobile Robot Based on ROS,* 37th CCC, 5207-5212 (2018)
13. D. Huang, Z. Cai, Y. Wang, X. He, *A real-time fast incremental SLAM method for indoor navigation,* CAC, **2013.6775723**, 171-176 (2013)
14. Y. Chen, J. Tang, C. Jiang, L. Zhu, M. Lehtomaki, H. Kaartinen, ¨ R. Kaijaluoto, Y. Wang, J. Hyyppa, H. Hyyppa, H. Zhou, ¨ L. Pei, R. Chen, *The Accuracy Comparison*

of Three Simultaneous Localization and Mapping (SLAM)-Based Indoor Mapping Technologies, Sensors, **18**, 10, 3228 (2018)

15. X. Du, K. K. K. Htet, K. K. Tan, *Development of a Genetic-Algorithm-Based Nonlinear Model Predictive Control Scheme on Velocity and Steering of Autonomous Vehicle,* IEEE Transaction on Industrial Electronics, **63**, 11, 6970-6977 (2016)
16. H. Wang, B. Liu, X. Ping, Q. An, *Path Tracking Control for Autonomous Vehicles Based on an Improved MPC*, IEEE Access, **7**, 161064-161073 (2019)
17. T.M. Vu, R. Moezzi, J. Cyrus, J. Hlava, *Model Predictive Control for Autonomous Driving Vehicles,* Electronics, **10**, 21, 2593 (2021)
18. R. Dai, W. Liu, Z. Bing, A. Knoll, *Variable Weight Model Predictive Contouring Control for Autonomous Tracking Based on Reinforcement Learning and Nonlinear Disturbance Observer,* IEEE 26th ITSC, 1563-1569 (2023)
19. J. Wang, F. Teng, J. Li, L. Zang, T. Fan, J. Zhang, *Intelligent vehicle lane change trajectory control algorithm based on weight coefficient adaptive adjustment*, Advances in Mechanical Engineering, **13**, 3 (2021)
20. W. Liu, M. Zeng, G. Chen and A. Knoll, *Deep High-Level Policy Model Predictive Contour Control for Autonomous Racing*, IEEE IV symposium, 1-7 (2023)
21. A. Heilmeier, A. Wischnewski, L.Hermansdorfer, J. Betz, M. Lienkamp, B. Lohmann, *Minimum curvature trajectory planning and control for an autonomous race car*, Vehicle system Dynamics, **58:10**, 1497-1527 (2020)