2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Curvature-Aware Model Predictive Contouring Control*

Lorenzo Lyons[1] and Laura Ferranti[1]

***Abstract*— We present a novel Curvature-Aware Model Predictive Contouring Control (CA-MPCC) formulation for mobile robotics motion planning. Our method aims at generalizing the traditional contouring control formulation derived from machining to autonomous driving applications. The proposed controller is able of handling sharp curvatures in the reference path while subject to non-linear constraints, such as lane boundaries and dynamic obstacle collision avoidance. Compared to a standard MPCC formulation, our method improves the reliability of the path-following algorithm and simplifies the tuning, while preserving real-time capabilities. We validate our findings in both simulations and experiments on a scaled-down car-like robot.**

I. INTRODUCTION

Automated vehicles, such as self-driving cars, have the potential to revolutionize our urban mobility. They can help us regulate traffic in big cities, reduce fatalities caused by human errors, and make transportation more accessible to elderly population or people with disabilities [1]. While we can gain a lot out of these technologies, the majority of commercial automated vehicles are still far from reaching the highest level of automation (i.e., SAE level 5) [2]. Safety is still a major concern for further developing these vehicles, especially in complex urban environments or in rural areas. While on the highway the vehicle is rarely required to perform harsh turns and avoid obstacles on narrow lanes, these situations often occur in our cities or in back-country roads. To navigate an autonomous vehicle in these challenging environments, the local motion planner plays a fundamental role since it acts as a bridge between how the vehicle perceives the environment and how it reacts to it. The motion planner has the task to keep the vehicle on the road, providing the right references to the actuators to stay within the lane boundaries, while avoiding collisions with obstacles. Precisely accounting for the road curvature is one of the challenges that the motion planner must face to keep the vehicle on track. This paper proposes a *curvature-aware* local motion planner based on nonlinear model predictive control (NMPC) to address these challenges.

Traditional approaches to path following include feedback controllers [3] or iterative linear quadratic regulator (iLQR) [4]. These approaches, however, consider decoupling lateral and longitudinal dynamics and/or do not account for state and actuator constraints. MPC is a valid alternative to these methods. MPC is an optimization-based control technique that allows one to formulate desired control objectives and optimize the behavior of the vehicle over a predefined time window (refer to [5] for an overview). A useful comparison among PID, iLQR and MPC can be found in [6].

In this paper, we build on a NMPC formulation suitable for path planning, known as model predictive contouring control (MPCC). Compared to reference-tracking MPC [7], MPCC includes an analytical description of the reference path, which is not parameterized with respect to time but with respect to a tailored path parameter. MPCC was originally applied in machining [8]–[10]. In this field path following had been addressed prior to these works, yet the focus was on providing geometrical convergence to the path by means of feedback controllers' (e.g., [11]). Understanding the origin of MPCC is relevant to the present work, since the formulation developed by [8] for machining has been transferred almost unaltered to the field of autonomous vehicles, for example in [12]–[14]. These approaches showed the potential of MPCC for robot navigation in dynamic environments. Classical MPCC, however, relies on the assumption of small lateral deviations of the vehicle from the reference path. This is surely the case in a high-precision application where the quality specifications often require lateral errors in the order of microns. In autonomous driving (and mobile robotics in general) this assumption may not hold since the presence of obstacles or actuator saturation may force the vehicle to deviate significantly from the reference path. Efforts to handle this issue can be found in the field of autonomous racing, where race tracks usually feature sharp turns. In [15] an effective strategy to account for high curvatures is presented, yet it is used offline to define an optimal path. The results are validated in simulation and path tracking is enforced by a low-level controller integrated in the simulation software. In [16] and [17] a similar approach is taken and the findings are validated on a full scale autonomous racing car. Exploiting the prior knowledge of the full track is a successful strategy in a racing context, yet this detailed knowledge is generally not available in an urban driving scenario. Handling tight curves relying only on local curvature information is the challenge addressed in the present work.

The main contribution of the present work is to develop a MPCC formulation for mobile robot navigation capable of accounting for sharp turns in, or deviating from, the reference path without any off-line computations. Thanks to our *curvature-aware* reformulation we show how the proposed MPCC method is more reliable (e.g., to keep the vehicle within road boundaries and avoid obstacles) and easier to tune than the traditional MPCC derived from machining. We support these claims both in simulation and with experiments

*This research is supported by the NWO-TTW Veni project HARMONIA (no. 18165).

[1]The authors are with the Reliable Robot Control Lab, Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands {l.lyons,l.ferranti}@tudelft.nl

on a real robotic platform.

The paper is structured as follows. Section II presents the problem formulation and details the limitations of traditional MPCC. Section III presents our method. Section IV compared the MPCC baseline with our method, both in simulation and in experiments. Finally, Section V concludes the paper.

II. PROBLEM FORMULATION

We consider a robot described by the following discretized dynamic equations:

$$\boldsymbol{x}(t+1) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^m$ are the state and control input of the vehicle, respectively. The state and the control input are subjected to limitations, that is, $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ (e.g., lane boundaries) and $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^m$ (e.g., steering and acceleration limits). The vehicle moves in a complex dynamic environment. In this respect, we consider dynamic obstacles moving with a constant velocity[1]. We represent the obstacles as circles with radius r_{obst} and with state $\boldsymbol{x}^{\text{o}}$. The set of neighboring obstacles is $\mathcal{O}$.

Our goal is to design a local motion planner for the ego vehicle to safely navigate in a tight dynamic environment efficiently, penalizing large deviations from the reference path $\boldsymbol{x}^{\text{ref}}$ provided by a high-level motion planner. This can be formalized as follows:

$$\min_{\boldsymbol{x}_k, \boldsymbol{u}_k} \sum_{k=0}^{N} J_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}^{\text{ref}}) \tag{2a}$$

$$\text{s.t.: Eq. (1)}, \tag{2b}$$

$$\boldsymbol{x} \in \mathcal{X},\ \boldsymbol{u} \in \mathcal{U}, \tag{2c}$$

$$||\boldsymbol{c}_k^i - \boldsymbol{x}_k^o||_2 \geq r + r_{\text{obst}}, i = 1, ..., n_{\text{disc}}, o \in \mathcal{O}, \tag{2d}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}}, \tag{2e}$$

where J indicates the planner objectives we aim to define and Constraint (2d) ensures collision avoidance between the neighboring obstacles and the center of each disc $\boldsymbol{c}^i$ of radius r used to represent the vehicle (following a similar strategy to [12]). Finally, $\boldsymbol{x}_{\text{init}}$ indicates the measurements the planner receives from the sensors at every sampling instant.

The problem above can be solved with MPCC, which is a well-established technique for motion planning. The remainder of the section provides a summary of the traditional MPCC formulation (Section II-A) and discusses its limitations (Section II-B).

A. Overview of MPCC

MPCC recursively solves Problem (2) with the cost J defined as follows:

$$J_k = q_1(v_k - v_t)^2 + q_2(\varepsilon_k^c)^2 + q_3(\varepsilon_k^{\text{lag}})^2 + \boldsymbol{u}_k^T Q_u \boldsymbol{u}_k \tag{3}$$

The cost is given by the sum of *(i)* three errors, that are, the deviation of v (i.e., the euclidean norm of the velocity of the mobile robot) w.r.t. the desired velocity v_t, the contour error ε^c and lag error ε^{lag} (defined below); and *(ii)* a quadratic penalty on the control action $\boldsymbol{u}$. The scalars q_1, q_2 and q_3, and the matrix Q_u are tuning parameters of the controller. The use of the lag and contour errors in the cost of the controller is what differentiates the MPCC controller from a standard NMPC for trajectory tracking. These errors are defined as follows:

$$\varepsilon^c = (\boldsymbol{p} - \boldsymbol{\phi}(s)) \cdot \hat{\boldsymbol{n}}(s),\ \varepsilon^{\text{lag}} = (\boldsymbol{p} - \boldsymbol{\phi}(s)) \cdot \hat{\boldsymbol{t}}(s),$$

where $\boldsymbol{p}$ is the position of the robot (i.e., $\boldsymbol{p} = [x\, y]^{\text{T}}$), $\boldsymbol{\phi}(s)$ is the reference path parameterized on the arc length s, $\hat{\boldsymbol{t}}(s)$ and $\hat{\boldsymbol{n}}(s)$ are respectively the tangent and normal unit vectors to the path at a given value of s, that is, they constitute the Frenet frame of reference [18]. We can thus interpret ε^c and ε^{lag} as attractive terms that pull $\boldsymbol{p}$ towards $\boldsymbol{\phi}(s)$.

The value of s is defined as $s = \text{argmin}_s\, ||\boldsymbol{p} - \boldsymbol{\phi}(s)||_2$. However, solving this minimization problem as a sub-problem of (2) results in a non-differentiable cost function. For this reason it is necessary to provide an expression of the dynamics of s, which are traditionally approximated as $\dot{s} \approx v$ and added to the dynamic Constraints (1).

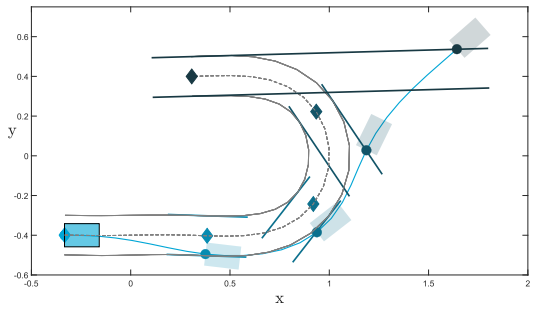
B. Limitations of MPCC

As stated in the introduction, the standard MPCC formulation was developed in the field of machining, where acceptable lateral deviation from the path are small, thus the *traditional* approximation $\dot{s} \approx v$ is reasonable. However, in the context of mobile robot navigation this approximation is not as easily verified, since large lateral deviations from the path may occur. When the assumption of $\dot{s} \approx v$ fails, two undesired behaviours may arise depending on the values of q_2 and q_3. The first undesired behavior is that since the predicted value of s is no longer accurate, any s-dependent constraints will be inaccurately evaluated. A relevant example in the context of autonomous driving are lane boundaries, that are often defined as bounds on ε^c [12], [19], namely $\varepsilon^c_{\text{min}} \leq \varepsilon^c(s) \leq \varepsilon^c_{\text{max}}$. The erroneous evaluation of the lane boundaries may lead the open-loop solution to violate these constraints without the solver returning any infeasibility errors. This issue is typical for low q_2 and q_3 gain values. In Figure 1(a) we show the open-loop solution obtained for a scenario in which a vehicle enters a tight curve. Note that the lane boundaries are expressed with respect to the vehicles' rear axle position (i.e., it is this point that should remain within the bounds). It is clear how the inaccurate s prediction induces the controller to violate the real lane constraints, yet it is very important to note that the solver converges to a feasible solution since according to it the lateral error remains within the bounds. This type of *failure* is particularly dangerous since it is not detectable by looking at the solver outcome.

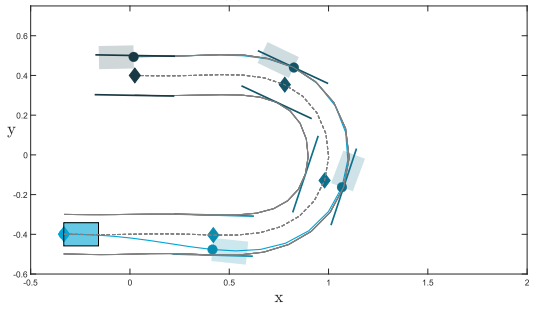
The second undesired effect is connected with the fact that $\boldsymbol{p}$ and s are closely related, as the following definition shows:

$$\boldsymbol{p}_k = \sum_{i=0}^{k} \boldsymbol{v}_i dt,\ s_k = \sum_{i=0}^{k} ||\boldsymbol{v}_i||_2 dt$$

[1]Our method can be interfaced with any perception algorithm able to provide predictions of the behavior of the obstacles. Predicting obstacles' behavior is outside the scope of this work.



(a) MPCC



(b) CA-MPCC

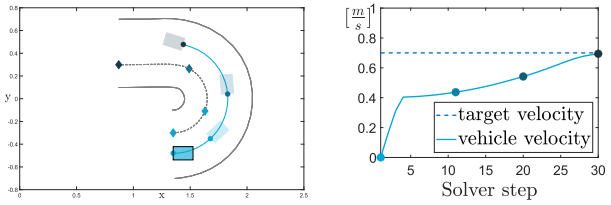
Fig. 1: Comparison of the open-loop trajectories. The dots are the vehicle rear axle positions for solver stages $1, 12, 18, 23, 30$ ordered from lighter to darker shades of blue, the diamonds are the corresponding $\boldsymbol{\phi}(s_k)$ as evaluated by the solver and the solid lines are the corresponding local lane boundaries.

When the vehicle is unable to closely follow the path (e.g., due to an obstacle), the optimal solution to Problem (2) may be to reduce the value of $\boldsymbol{v}$ in order to reduce $\boldsymbol{p} - \boldsymbol{\phi}(s)$, and in turn ε^c and ε^{lag}. This effect is more evident for higher q_2 and q_3 gains. Figure 2(a) shows the open-loop solution in a scenario in which the vehicle enters a turn with a certain amount of initial lateral error. Steering limitations force the vehicle to keep a certain distance from the path, as a result we can see how $\boldsymbol{\phi}(s)$ is wrongly evaluated. Figure 2(b) shows the open-loop velocity profile. Note that the vehicle starts from a stationary condition. We can see how in the first stages the algorithm fails to track the velocity, only to recover once the vehicle exits the curve. Notice that both limitations can lead to dangerous driving scenarios. In addition, the occurrence of these problems is highly dependent on the tuning of the controller. This makes the tuning of the controller extremely challenging and scenario dependent. Our method aims to solve these limititations, as discussed in the next section.

III. CURVATURE-AWARE MPCC

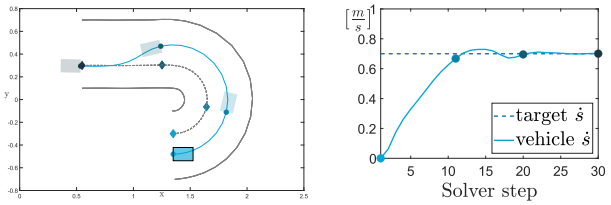
Accurately evaluating $\dot{s}$ is key to overcome the issues listed in the Sec. II-B. The analytical expression of $\dot{s}$ is:

$$\dot{s} = \frac{\boldsymbol{v} \cdot \hat{\boldsymbol{t}}(s)}{1 - \kappa(s)\varepsilon^c}, \tag{4}$$



(a) (b)

Baseline MPCC



(c) (d)

Curvature-Aware MPCC

Fig. 2: Plots (a) and (c) show the open-loop solutions. Plots (b) and (d) show the velocity and $\dot{s}$ profile, respectively. The dots match the predictions plotted in (a) and (c).

where $\kappa(s)$ is the curvature of the path. As described in [15] this expression can be obtained by re-writing $\boldsymbol{p}$ in the $\hat{\boldsymbol{t}}$ and $\hat{\boldsymbol{n}}$ frame and taking the first time derivative. Using Eq. (4) directly in Problem (2), however, comes with a significant increase in computation time required for online optimization. This is because $\hat{\boldsymbol{t}}$ and $\kappa(s)$ have to be evaluated starting from an analytical expression of $\boldsymbol{\phi}(\boldsymbol{s})$ that is generally computationally expensive. For this reason we provide a lightweight approximation of $\dot{s}$. According to Eq. (4) $\dot{s}$ can be interpreted as the projection of $\boldsymbol{v}$ on $\hat{\boldsymbol{t}}$ times a curvature-dependent scaling ratio. We can define the scaling ratio σ as:

$$\sigma = \frac{1}{1 - \kappa(s)\varepsilon^c} = \frac{R(s)}{R(s) - \varepsilon^c},$$

where we have rewritten κ as $\frac{1}{R}$, R is the curvature radius. Let $\boldsymbol{C}$ be the centre of curvature of the path. We consider an infinitesimal change in position $d\boldsymbol{p} = \boldsymbol{v}dt$ and we denote the corresponding infinitesimal change in the path parameter as derived from Eq. (4) as ds^*. Let the point $\tilde{\boldsymbol{p}}_t$ and $\tilde{\boldsymbol{s}}$ be defined respectively as:

$$\tilde{\boldsymbol{p}}_t = \boldsymbol{p} + \boldsymbol{v}dt \cdot \hat{\boldsymbol{t}},$$
$$\tilde{\boldsymbol{s}} = \boldsymbol{\phi}(s) + \hat{\boldsymbol{t}}ds^*.$$

We can now give a geometrical interpretation to Eq. (4) by noticing that the length of the segments $\overline{\boldsymbol{C}\boldsymbol{\phi}(s)}$ and $\overline{\boldsymbol{C}\boldsymbol{p}}$ are respectively R and $R - \varepsilon^c$. Since the triangles $_{\Delta}\boldsymbol{C}\boldsymbol{p}\tilde{\boldsymbol{p}}_t$ and $_{\Delta}\boldsymbol{C}\boldsymbol{\phi}(s)\tilde{\boldsymbol{s}}$ are similar we can write:

$$\frac{ds^*}{v_t dt} = \frac{||\tilde{\boldsymbol{s}} - \boldsymbol{\phi}(s)||_2}{||\tilde{\boldsymbol{p}} - \boldsymbol{p}_t||_2} = \frac{R}{R - \varepsilon^c} = \sigma,$$

where for ease of notation purposes we have defined $v_t = \boldsymbol{v} \cdot \hat{\boldsymbol{t}}$ and dropped the $R(s)$ dependency. From this expression we

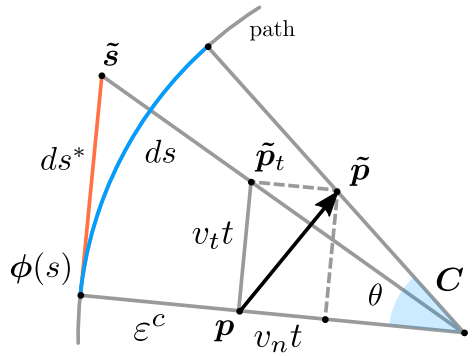


Fig. 3: Geometrical interpretation of the traditional (orange) and proposed (blue) $\dot{s}$ formulation.

can see how σ can be interpreted as a projection ratio, see Figure 3. This geometrical interpretation highlights how even the analytical expression of $\dot{s}$ becomes inaccurate for large Δt values and high path curvature. We thus propose a novel $\dot{s}$ expression that is exact under the assumptions of constant curvature radius and robot velocity, and is therefore a suitable approximation for large Δt values when curvature and robot velocity vary smoothly. To simplify notation (without loss of generality) we assume that $t_0 = 0$. In this way we can rewrite Δt as t. We also define the point $\tilde{\boldsymbol{p}} = \boldsymbol{p} + \boldsymbol{v}t$. The proposed method relies on first evaluating the angle θ between $\boldsymbol{p}$, $\boldsymbol{C}$ and $\tilde{\boldsymbol{p}}$ and then evaluating ds as $ds = R\theta$. Defining $v_n = \boldsymbol{v} \cdot \hat{\boldsymbol{n}}$, as shown in Figure 3, the angle θ can be calculated as:

$$\theta = \arctan\left(\frac{v_t t}{R - \varepsilon^c - v_n t}\right) \tag{5}$$

To obtain $\dot{s}$ we simply take the derivative of $ds = R\theta$ with respect to t, that is:

$$\dot{s}(t) = R\frac{v_t(R - \varepsilon^c - v_n t) + v_t v_n t}{(R - \varepsilon^c - v_n t)^2 + (v_t t)^2} \tag{6}$$

Notice that for $t \to 0$ Eqs. (4) and (6) coincide. It is also important to highlight that since v_t and v_n are fixed, (6) is the time derivative of s for a point moving in a constant direction. Expressions (4) and (6) have similar computational complexity since they both require the evaluation of $\hat{\mathbf{n}}(\mathbf{s})$ and $\hat{\mathbf{t}}(\mathbf{s})$ to obtain R, v_t and v_n. However we propose to use Eq. (6) in its integral form, that is, $ds = R\theta$, for the whole integration step. After evaluating $d\boldsymbol{p}$ by means of any integration scheme, we evaluate $v_t t$ and $v_n t$ as $d\boldsymbol{p} \cdot \hat{\boldsymbol{t}}$ and $d\boldsymbol{p} \cdot \hat{\boldsymbol{n}}$, respectively, and use them in Eq. (5) to evaluate $ds = R\theta$. Notice that by construction (6) has the property $\int_0^t \dot{s} dt = R\theta$. This approach has similar computational cost as using (4) in a Forward Euler method, yet it achieves much higher accuracy for small R and large t values, as can be seen in Figure 3. Another advantage of this formulation is that the integration scheme used to evaluate the system dynamics ($d\boldsymbol{p}$) is decoupled from the one used to evaluate ds, since the latter is evaluated after the former. This also implies that any dynamic model can be chosen, as long as it provides the change in position of a chosen reference point on the vehicle.

The new definition of $\dot{s}$ in (6) can be used to derive the following Curvature-Aware MPCC formulation, namely a CA-MPCC, suitable for generic motion planning applications:

$$\min_{\boldsymbol{u}_k} \sum_{k=0}^{N} q_1(\dot{s}_k - \dot{s}_t)^2 + q_2(\varepsilon_k^c)^2 + \boldsymbol{u}_k^T Q_u \boldsymbol{u}_k \tag{7a}$$

$$\text{s.t.} \quad \begin{bmatrix} \boldsymbol{x}_k \\ s_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) \\ s_{k-1} + R_{k-1}\theta_{k-1} \end{bmatrix} \tag{7b}$$

$$\text{Eq. (2d), (2c), (2e),} \tag{7c}$$

where R_{k-1} and θ_{k-1} are the respective quantities defined for time instant $k-1$. Equation 7b is also suitable for straight paths, since for $R \to \infty$, $R\theta = R\arctan\frac{v_t t}{R - \varepsilon^c - v_n t} \to v_t t$. Notice that the definition of the cost differs from Eq. (3). We replace the velocity and lag errors' penalties with an $\dot{s}$ tracking term. This is now possible since thanks to (6) we have access to this quantity. Notice that in the cost function, $\dot{s}_k$ is required. This is equivalent to taking the limit for $t \to 0$ in Eq. (6), that is equivalent to (4) as previously pointed out. It also worth mentioning that if the application requires a trade-off between v and $\dot{s}$ tracking, it is straightforward to add the velocity tracking term back into the cost function. We were also able to eliminate the lag error term since $\dot{s}$ also carries information on the heading direction of robot relative to the path. Having removed such an additional cost term simplifies the tuning of the CA-MPCC, as further discussed in Section IV-A.

Figure 1(b) shows the open-loop solution obtained with our CA-MPCC in the same conditions and same q_1, q_2, Q_u as for the baseline MPCC. Notice how, compared to the baseline MPCC, the lane boundaries are correctly evaluated thanks to a more precise s prediction. As a result the open-loop solution does not violate the lane boundary constraints. Figure 2(c) shows the solution to an initial standstill condition featuring some initial lateral error. Steering limitations do not allow the solver to reduce the initial lateral error, yet, as Figure 2(d) shows, the open-loop solution can accurately track the desired $\dot{s}$, leading to a safer driving experience.

IV. RESULTS

This section compares the closed-loop performance of the baseline- and CA-MPCC formulations.

Model. We tested the method on an autonomous driving application. The model used by both methods (i.e., in (2b) and (7b)) is given by the following equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\eta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos\eta \\ v\sin\eta \\ \frac{v\tan(\delta)}{l} \\ -cv + \alpha\tau - \beta \end{bmatrix}$$

where the states x, y, η and v are respectively the position of the rear axle, orientation, and longitudinal velocity of the vehicle. The inputs τ and δ are throttle and steering, respectively; l is the length between the front and rear axle of the robot. The longitudinal acceleration is defined as

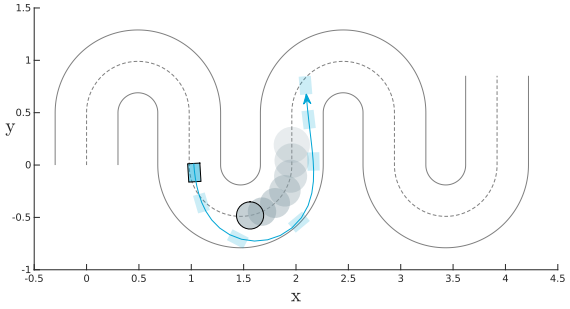


Fig. 4: Track used for parameter sweep. The solid gray circle is the dynamic obstacle, while the faded gray circles are the dynamic obstacle's predicted positions.

TABLE I: Overview of the tuning parameters used for the baseline MPCC and for the CA-MPCC.

Weights sweep parameters		
	MPCC	CA-MPCC
q_2	$q_2 = i,\ i \in \{0, ..., 11\}$	$q_2 = 0.2i,\ i \in \{0, ..., 25\}$
q_3	$q_3 = 0.5i,\ i \in \{0, ..., 11\}$	
Trials per data point	20	20
Other parameters		
Problem	Vehicle	Obstacle
$v_t = 0.75$ m/s (MPCC)	$\delta_{max} = 20°$	$r_{obst} \in [0.10, 0.15]$ m
$\dot{s}_t = 0.75$ m/s (CA-MPCC)	$l = 0.175$ m	$\dot{r}_{obst} \in [0.01, 0.02]$ m
$R_{\text{track}} = 0.5$	$R_{\text{max}} = 0.48$ m	$\dot{s}_{obst} \in [0.2, 0.3]$ m/s
$\varepsilon^c_{\text{max,min}} = \pm 0.3$ m		

$\dot{v} = -cv + \alpha\tau - \beta$, where c is the damping coefficient, and α and β are motor coefficients. We evaluated their values experimentally through step-response tests on the real platform that will be used for the experiments (Section IV-B). Actuator limitations are defined as $\tau_{\min} \leq \tau_k \leq \tau_{\max}$ and $\delta_{\min} \leq \delta_k \leq \delta_{\max}$.

Local Path. Both methods require a strategy to evaluate $\boldsymbol{\phi}(s)$. For this, we decided to describe the local path with Chebyshev polynomials [20]. We split $\boldsymbol{\phi}(s)$ into its x and y components, and perform the respective polynomial fits. This fitting operation is repeated at each control loop.

We also add a terminal cost term $\boldsymbol{x}_N^T Q_P \boldsymbol{x}_N$ in (2a) and (7a), which approximates the cost-to go from $k = N$ to $k = +\infty$ and thus ensures that the closed-loop behaviour resembles the open-loop predictions [21].

In the remainder of the section, we test the baseline MPCC and our CA-MPCC both in simulation and with real-life experiments on an autonomous driving scenario.

A. Simulations

To compare the reliability and the ease of tuning of the two approaches we tasked the algorithms with guiding an autonomous vehicle through a racetrack at a desired speed. The racetrack features some sharp turns that almost match the steering capabilities of the vehicle and some slow-moving dynamic obstacles that need to be overtaken. Figure 4 shows the chosen track. We have subsequently performed a parameter sweep on the weights. For both methods and for all tests, we kept $q_1 = 1$, and $q_\tau = 0.1$, $q_\delta = 0.1$, where q_τ and q_δ weight the respective inputs. Since $q_1 = 1$ the other weights are normalized with respect to the latter. Table I indicates the relevant parameters for the tests. R_{track} is the radius of the curves in the track, while R_{max} is the maximum curvature radius of the vehicle. The dynamic obstacle moves along the path at a speed $\dot{s}_{\text{obst}}$, and we simulate uncertainty in its occupancy predictions by increasing its radius at a rate $\dot{r}_{\text{obst}}$ along the prediction horizon. Once a dynamic obstacle is overtaken, it is re-positioned ahead of the mobile robot, each time randomly drawing new obstacle parameters from the sets described in Table I. A test is considered to be a success if no lane boundary violation occurs, and if the vehicle reaches the end of the track within a given time interval. This time interval is chosen such that if the average velocity of the robot was less than 60% of v_t or $\dot{s}_t$, the test results in a failure. This occurs if the vehicle is unable to overtake the dynamic obstacle.

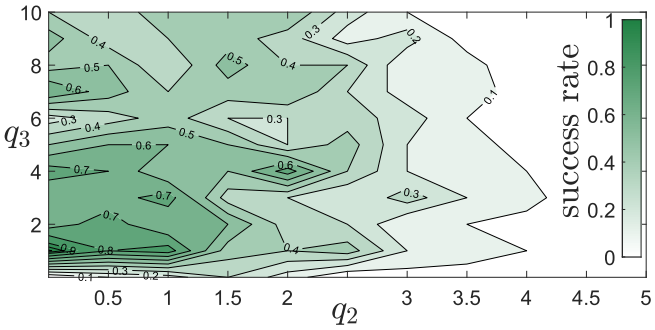


Fig. 5: Baseline MPCC parameter sweep.

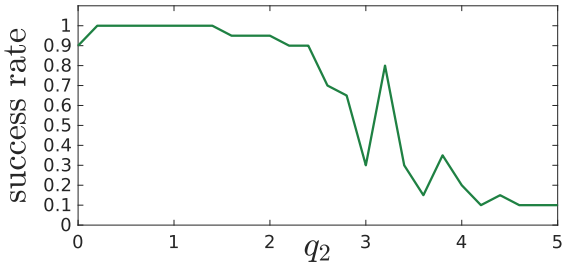
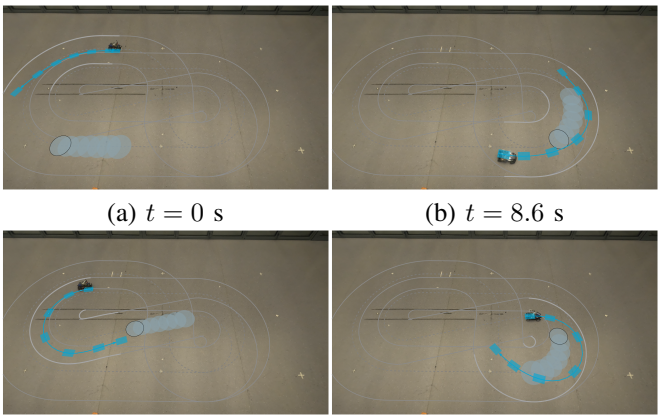


Fig. 6: Curvature-Aware MPCC parameter sweep.

Figure 5 shows the results for the baseline MPCC. As evident from the figure, the tuning of the baseline MPCC can be extremely challenging. In contrast, Figure 6 shows the results for the Curvature-Aware MPCC. This figure shows that below a certain threshold of q_2, the behaviour of the closed loop system is very reliable, since the success rate is close or equal to 100%. Over this threshold the success rate gradually decreases. The failures are entirely due to the vehicle no longer being able to overtake the dynamic obstacle. This behaviour is to be expected for very large ε^c weights. These results support our claim that the CA-MPCC is easier to tune, since it features one less parameter in the cost function, and the closed-loop behaviour is much more consistent within a certain q_2 range. Furthermore, this range can be conservatively estimated by keeping $q_2 \leq q_1$ (in the



(a) $t = 0$ s (b) $t = 8.6$ s

(c) $t = 16.6$ s (d) $t = 22.4$ s

Fig. 7: CA-MPCC performing the circuit at different time instants.

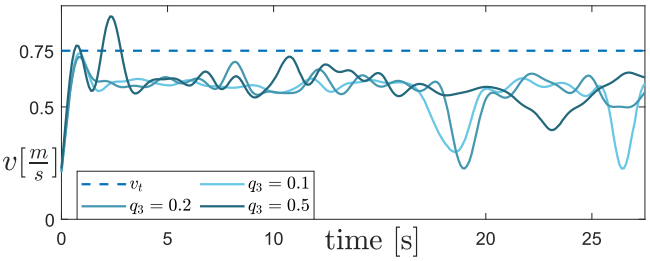


Fig. 8: Velocity profile during baseline MPCC experiments.

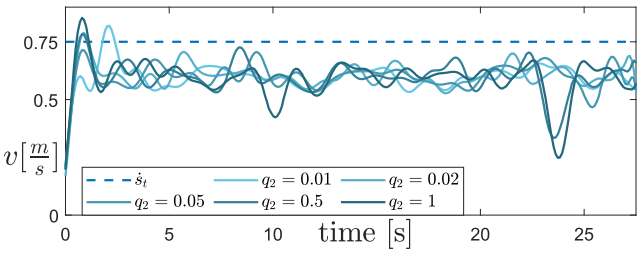


Fig. 9: $\dot{s}$ profile during CA-MPCC experiments.

tests the weights are normalized with respect to q_1). This would generally be the case in real life application since lower lateral error gains result in smoother robot trajectories.

B. Experiments

We validate our findings on a real mobile robot, a 1:10 scaled-down car-like vehicle, a modified Waveshare JetRacer Pro AI kit. Since localization is not the focus of the present work, we relied on a Motive OptiTrack motion capture system. To control the vehicle a dedicated ROS network was set up. The navigation algorithm runs on a laptop featuring an AMD Ryzen 7 5800H processor. As for the simulations we task both the baseline MPCC and the CA-MPCC with driving along a track that features sharp turns while overtaking a slow-moving virtual dynamic obstacle. Figure 7 shows the CA-MPCC performing a circuit lap.

We fixed the lateral error weight $q_2 = 0.5$ for the baseline MPCC. The value was selected after some trial-error experiments to achieve a consistent vehicle behavior throughout all the experiments. Figure 8 examines the effects of different q_3 weights on the velocity tracking error of the MPCC. For low q_3 values the velocity drops significantly during some turns (t $\approx 18s, 27s$). This is due to the error in lane boundary evaluation, as anticipated in Section II-B. During the whole experiment the solver converged to a seemingly valid solution, making this kind of failure difficult to identify. For higher q_3 weights this behaviour becomes progressively less frequent and is absent in the final test. However, a drop in velocity is still present in the tightest bend of the circuit (t $\approx 23s$). This behaviour is due to the lag error term being too large compared to the velocity tracking term. The open-loop solutions are similar to those presented in Figures 1(a) and 2(a) for low q_3 and high q_3 respectively, as can be seen in the video attached to this submission. To test the CA-MPCC algorithm we ran some tests changing the lateral error weight. Figure 9 shows the $\dot{s}$ tracking performance during the experiments. We notice drops in $\dot{s}$ in the experiments featuring $q_2 = 0.5$ and $q_2 = 1$ (t $\approx 24s$), yet this failure is due to modelling errors in the system dynamics, rather than to the control algorithm. This can be seen from the difference between the open-loop and closed-loop behaviour as is evident from the video footage [22]. In both sets of experiments we observe a constant bias in the velocity or $\dot{s}$ tracking, this is likely due to a modelling error in the correlation between τ and $\dot{v}$ (which affects the experiments with both algorithms). Compared to the standard MPCC formulation we notice that the closed-loop behaviour is much more consistent, even for very large parameter variations, this is in accordance with the simulations.

Concerning the computation times, we measured the full control loop (i.e. including the polynomial fitting operation, which takes on average 3 ms). Over all the MPCC experiments the mean, 95% percentile, and the worst-case are 19.7, 28.2, and 44.2 ms, respectively. For the CA-MPCC we measured 28.6, 52.3, and 65.0 ms, respectively. The CA-MPCC does feature higher computation times, yet is still achieves real-time feasibility. It is important to mention that it may be possible to reduce computation time by tuning the number of Chebyshev polynomial bases that represent the local path. For our experiments we used 20 polynomial bases, yet 10 could also have been sufficient.

V. CONCLUSIONS

This work presented an online Curvature-Aware MPCC formulation capable of handling sharp turns in the reference path, while avoiding collisions with dynamic obstacles. We compared our method against a baseline MPCC formulation and proved in both simulation and experiments that the proposed method is both easier to tune and more reliable. Future research directions are to extend this method to 3D motion planning and racing where no previous knowledge of the racetrack is available, yet effort should be directed towards further reducing computation times.

REFERENCES

[1] N. Lang, A. Herrmann, M. Hagenmaier, and M. Richter, "Can Self-Driving Cars Stop the Urban Mobility Meltdown?" 8 2020. [Online]. Available: https://www.bcg.com/publications/2020/how-autonomous-vehicles-can-benefit-urban-mobility

[2] "The 6 Levels of Vehicle Autonomy Explained." [Online]. Available: https://www.synopsys.com/automotive/autonomous-driving-levels.html

[3] Yan Ding, "Three Methods of Vehicle Lateral Control: Pure Pursuit, Stanley and MPC," 3 2020. [Online]. Available: https://dingyan89.medium.com

[4] A. Nagariya and S. Saripalli, "An iterative lqr controller for off-road and on-road vehicles using a neural network dynamics model," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1740–1745.

[5] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: an engineering perspective," pp. 1327–1349, 11 2021.

[6] S. He, J. Zeng, and K. Sreenath, "Autonomous racing with multiple vehicles using a parallelized optimization with safety guarantee using control barrier functions," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 3444–3451.

[7] N. Chowdhri, L. Ferranti, F. S. Iribarren, and B. Shyrokau, "Integrated nonlinear model predictive control for automated driving," *Control Engineering Practice*, vol. 106, p. 104654, 2021.

[8] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *Proceedings of the IEEE Conference on Decision and Control*. Institute of Electrical and Electronics Engineers Inc., 2010, pp. 6137–6142.

[9] ——, "Application of model predictive contouring control to an XY table," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 10 325–10 330, 2011.

[10] A. Rupenyan, M. Khosravi, and J. Lygeros, "Performance-based trajectory optimization for path following control using bayesian optimization," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 2116–2121.

[11] R. Skjetne, T. I. Fossen, and P. V. Kokotović, "Robust output maneuvering for a class of nonlinear systems," *Automatica*, vol. 40, no. 3, pp. 373–383, 3 2004.

[12] W. Schwarting, J. Alonso-Mora, L. Pauli, S. Karaman, and D. Rus, "Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention," in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., 7 2017, pp. 1928–1935.

[13] O. De Groot, B. Brito, L. Ferranti, D. Gavrila, and J. Alonso-Mora, "Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5389–5396, 7 2021.

[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. Rucco, G. Notarstefano, and J. Hauser, "An efficient minimum-time trajectory generation strategy for two-track car vehicles," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1505–1519, 7 2015.

[16] J. L. Vazquez, M. Bruhlmeier, A. Liniger, A. Rupenyan, and J. Lygeros, "Optimization-based hierarchical motion planning for autonomous racing," in *IEEE International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers Inc., 10 2020, pp. 2397–2403.

[17] S. Srinivasan, S. N. G. Nicolas Giles, and A. Liniger, "A Holistic Motion Planning and Control Solution to Challenge a Professional Racecar Driver," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7854–7860, 10 2021.

[18] A. Micaelli and C. Samson, "Trajectory tracking for unicycle-type and two-steering-wheels mobile robots," Ph.D. dissertation, INRIA, 1993.

[19] L. Ferranti, B. Brito, E. Pool, Y. Zheng, R. M. Ensing, R. Happee, B. Shyrokau, J. F. P. Kooij, J. Alonso-Mora, and D. M. Gavrila, "SafeVRU: A research platform for the interaction of self-driving vehicles with vulnerable road users," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1660–1666.

[20] J. C. Mason and D. C. Handscomb, *Chebyshev polynomials*. Chapman and Hall/CRC, 2002.

[21] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[22] Lorenzo Lyons and Laura Ferranti, "Curvature-Aware Model Predictive Contouring Control," 2023. [Online]. Available: https://www.youtube.com/watch?v=6-E3I99D2sc