MPC-Based Automated Evasive Manoeuvres for Electric Remove Control Vehicles

Filipa Ribeiro
filipa.sousa.ribeiro@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2020

Abstract

The main goal of this thesis consists of converting an electric remote control (RC) vehicle into a fully autonomous vehicle. To this end, a Nonlinear Model Predictive Control (NMPC) approach for simultaneous path planning, path tracking, and obstacle avoidance is presented. NMPC has the capacity of dealing with system nonlinearities, foresee the future evolution of a process, and control it while fulfilling a set of constraints. The current work uses a nonlinear dynamic bicycle model with the Pajecka's tire model to predict the RC vehicle states throughout the prediction horizon. The path planning, path tracking, and obstacle avoidance form a nonlinear optimization problem that is solved in real time and during navigation of the RC vehicle (online). To implement a real-time NMPC scheme on a low-cost embedded system, an efficient nonlinear programming (NLP) solver is required. In the scope of this work, two different NLP solvers are investigated, which are the gradient search method and the state-of-the-art solver IPOPT. The simulation results show that both solvers present similar solutions. However, the gradient search method presents faster computational times. Furthermore, two local path planners, weighted distance and modified parallax method, are compared through simulation. The results demonstrate that the modified parallax method is more effective in obstacle avoidance maneuvers than distance-based methods, especially in cluttered environments. The performance of the purposed controller is experimentally tested on an RC vehicle. The experimental results prove that the gradient search method is a suitable NLP solver for real-time NMPC implementations in simple embedded systems with limited computational power. The modified parallax method has proven itself superior to the weighted distance approach, as it allowed the RC vehicle to avoid obstacles on its path to the destination point.

Keywords: Model Predictive Control, Path Planning, Obstacle Avoidance, Potential Functions, Autonomous Driving

1. Introduction

The last two decades have seen a growing trend towards the development of autonomous vehicles due to their wide range of applications, for example, in the automotive, truck, public transport, industrial, and military fields [5]. Intelligent vehicles can lower fuel consumption, reduce pollution rates [15], and prevent vehicle accidents [13].

In general, a fully autonomous vehicle must carry the capacity to perceive the local environment, such as recognize static and dynamic obstacles [6], and to choose the most adequate collision-free trajectory. Recently, researchers have shown an increased interest in solving the path tracking, obstacle avoidance and path planning problems using Model Predictive Control (MPC) techniques [22]. This is because MPC is currently considered the most powerful and complete approach for constrained control problems [11]. However, it comes at the expense of high computational demand, especially in nonlinear setups, *i.e.*, Nonlinear MPC (NMPC).

The study of real-time feasibility of NMPC-based steering control on a low-cost embedded system was conducted by Quirynen *et. al.* (2018) [18]. To prove the concept, the authors used a Raspberry Pi 2 and the ACADO code generation tool. The results reinforced the importance of choosing adequately the vehicle model, nonlinear programming (NLP) solver, and control horizon length to obtain the computations in real time. Liniger *et. al.* (2015) [12] implemented hierarchical and one-level NMPC in RC vehicles. The authors performed linearization of the vehicle model to build local approximations of the control NLPs in the form of convex quadratic programs. Despite the recent successful results in NMPC applied to low-cost embedded systems such as RC vehicles, most of its research is still based on linear MPC implementations [10].

The current thesis focuses on developing an NMPC algorithm for real-time path planning, obstacle avoidance and path tracking. First, this problem is studied in simulation and, afterwards, experiments on a remote control (RC) vehicle are performed in order to validate the proposed controller.

2. Vehicle Model

NMPC algorithms require a dynamical representation of the control system in order to foresee its future evolution. This section derives an analytical model of the RC vehicle dynamics and its tires. At the end, the vehicle model is validated.

2.1. Dynamical Model of the Experimental Vehicle

The nonlinear dynamical bicycle model captures key aspects of vehicle dynamics, since it accounts for the forces that act on the front and rear tires. Therefore, it represents more accurately the tire/ground interaction and phenomenons as side and longitudinal slip. The nonlinear model is transformed into a more computationally efficient model without loss of key dynamic characteristics of the vehicle. Consequently, the resulting simplified dynamical bicycle model is given by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} -\dot{\varphi} + \frac{2F_{\mathrm{s,f}}+2F_{\mathrm{s,r}}+2F_{\mathrm{l,f}}\mathbf{u}}{mv} \\ \dot{\varphi} \\ \frac{2L_{\mathrm{f}}F_{\mathrm{s,f}}-2L_{\mathrm{r}}F_{\mathrm{s,r}}+2L_{\mathrm{f}}F_{\mathrm{l,f}}\mathbf{u}}{I_{\mathrm{zz}}} \\ v\cos\varphi - v\beta\sin\varphi \\ v\sin\varphi + v\beta\cos\varphi \end{bmatrix}, \tag{1}$$

where m denotes the mass, L_{f} and L_{r}, correspond to the lengths from the CoG to the front and rear wheels, I_{zz} is the inertial moment relative to the Z-axis and v is the velocity of the vehicle. The state and control input vectors are $\mathbf{x} = [\beta, \varphi, \dot{\varphi}, X, Y]^T$ and $\mathbf{u} = \delta_{\mathrm{f}}$. In particular, β is the side-slip angle, φ symbolizes the yaw angle, $\dot{\varphi}$ denotes the yaw angular rate, and X and Y are the Cartesian coordinates in the inertial frame. Finally, δ_{f} corresponds to the front steering angle. The dimensions of the state and input vectors are n_{x} and n_{u}, respectively.

The output vector is given by

$$\eta = h(\mathbf{x}) = [X, Y]^T, \tag{2}$$

since X and Y are the desired control variables.

The longitudinal and lateral tire forces in the tire frame, *i.e.*, $F_{\mathrm{l,f}}$, $F_{\mathrm{l,r}}$ and $F_{\mathrm{s,f}}$, $F_{\mathrm{s,r}}$, are nonlinear functions that depend on the slip angle α, slip ratio s, total vertical load F_{z}, and friction coefficient μ between the tire and the road. It is assumed that the front and rear vertical loads, $F_{\mathrm{z,f}}$ and $F_{\mathrm{z,r}}$, are constant and that their distribution depends on the geometry of the vehicle. They are computed as

$$F_{\mathrm{z,f}} = \frac{L_{\mathrm{r}}\, mg}{2(L_{\mathrm{f}}+L_{\mathrm{r}})}, \qquad F_{\mathrm{z,r}} = \frac{L_{\mathrm{f}}\, mg}{2(L_{\mathrm{f}}+L_{\mathrm{r}})}, \tag{3}$$

with g being the gravitational acceleration.

The slip angle of both front and rear tires, α_{f} and α_{r}, are given by

$$\alpha_{\mathrm{f}} = \tan^{-1}\left(\frac{(\dot{y}+L_{\mathrm{f}}\,\dot{\varphi})\cos\delta_{\mathrm{f}} - \dot{x}\sin\delta_{\mathrm{f}}}{(\dot{y}+L_{\mathrm{f}}\,\dot{\varphi})\sin\delta_{\mathrm{f}} + \dot{x}\cos\delta_{\mathrm{f}}}\right), \tag{4}$$

$$\alpha_{\mathrm{r}} = \tan^{-1}\left(\frac{\dot{y}+L_{\mathrm{r}}\,\dot{\varphi}}{\dot{x}}\right), \tag{5}$$

The slip ratio of both front and rear tires, s_f and s_r, are defined as

$$s_\star = \begin{cases} \frac{rw_\star}{v_{\mathrm{l},\star}} - 1, & \text{if } v_{l,\star} > rw_\star, v_{\mathrm{l},\star} \neq 0 \quad \text{for breaking} \\ 1 - \frac{v_{\mathrm{l},\star}}{rw_\star}, & \text{if } v_{l,\star} < rw_\star, w_\star \neq 0 \quad \text{for driving} \end{cases} \tag{6}$$

where w is the angular velocity of the tire and r is the effective wheel radius. The subscript $\star$ either denotes the front or rear wheel {f,r}.

All parameter values of the RC vehicle are presented in Table 1, with I_{zz} being estimated in [7].

Parameter	Value	Unit
m	2.77	kg
I_{zz}	0.015	$\mathrm{kg \cdot m^2}$
L_{f}	0.12	m
L_{r}	0.14	m
L	0.365	m
W	0.21	m
$F_{\mathrm{z,f}}$	7.3160	N
$F_{\mathrm{z,r}}$	6.2709	N

Table 1: Parameter values of the RC vehicle.

2.2. Tire Model

In a dynamic driving scenario, in which a vehicle tries to avoid an obstacle through an evasive lateral maneuver, accurate estimations of the lateral forces are required. For this reason, this thesis uses the Pacejka's Magic Formula presented in [16], which is a high fidelity tire model. The Magic Formula describes the characteristics of the side and brake forces with good accuracy and physical meaning.

2.3. Model Validation

The simplified dynamical bicycle model presented in (1) is compared with the fully nonlinear dynamic bicycle model, for a constant vehicle speed of $v = 1\,\mathrm{m/s}$. The root-mean-square error (RMSE) is used to measure the error between the two models as follows

$$\mathrm{RMSE_j} = \sqrt{\sum_{i=0}^{\mathcal{M}} \frac{(\hat{\mathbf{x}}_{\mathrm{j,i}} - \mathbf{x}_{\mathrm{j,i}})^2}{\mathcal{M}+1}}, \tag{7}$$

where $\mathbf{x}_{\mathrm{j,i}}$ and $\hat{\mathbf{x}}_{\mathrm{j,i}}$ represent any element of the state vector $\mathbf{x}$ of the fully nonlinear and simplified dynamical bicycle models, at time step i, in the simulation horizon $\mathcal{M}$.

Table 2 presents the obtained RMSE and maximum absolute error values. The Y state presents the highest RMSE and absolute error of approximately 3.5 cm and 5 cm. Nevertheless, these values are still low and, therefore, the difference between the simplified and fully nonlinear models is negligible. For this reason, the simplified dynamical bicycle model in (1) is used in the NMPC implementation on the experimental vehicle.

States	RMSE	Max error
β	0.0035°	0.0052°
φ	0.0149°	0.0247°
$\dot{\varphi}$	0.0288 °/s	0.0503 °/s
X	0.0069 m	0.01 m
Y	0.0351 m	0.0478 m

Table 2: RMSE and maximum absolute state error values of the simplified dynamical bicycle model.

3. Model Predicted Path Planning

The hierarchical approach is typically considered in autonomous vehicles, which means that the path planning, obstacle avoidance and path tracking problems are solved separately [9]. Nevertheless, this thesis designs an NMPC algorithm that combines those problems into a single NLP problem.

3.1. NMPC Formulation

At time t, given the measurement of the current state vector $\hat{\mathbf{x}}_{\mathrm{t}}$, the NLP solver computes the optimal control sequence $\mathbf{u}_{\mathrm{t}}(\cdot)^*$ that solves the following NLP problem

$$\min_{\mathbf{x}_{\mathrm{t}}(\cdot),\mathbf{u}_{\mathrm{t}}(\cdot)} \Phi(\mathbf{x}_{\mathrm{N,t}}) + \sum_{k=0}^{N-1} L(\mathbf{x}_{\mathrm{k,t}},\mathbf{u}_{\mathrm{k,t}}) + PF_{\mathrm{goal,k,t}} + PF_{\mathrm{obs,k,t}} \tag{8a}$$

$$\text{s.t.} \quad \mathbf{x}_{\mathrm{0,t}} - \hat{\mathbf{x}}_{\mathrm{t}} = 0\,, \tag{8b}$$

$$\mathbf{f}^{\mathrm{dt}}(\mathbf{x}_{\mathrm{k,t}},\mathbf{u}_{\mathrm{k,t}}) - \mathbf{x}_{\mathrm{k+1,t}} = 0\,, \tag{8c}$$

$$|\mathbf{u}_{\mathrm{k,t}}| - u_{\mathrm{sat}} \le 0\,, \tag{8d}$$

$$|\alpha_{\mathrm{f,k,t}}| - \alpha_{\mathrm{f,sat}} \le 0\,, \tag{8e}$$

$$|\alpha_{\mathrm{r,k,t}}| - \alpha_{\mathrm{r,sat}} \le 0\,, \tag{8f}$$

where $k = 0, ..., N-1$ and N is the prediction horizon. Furthermore, at time t, the predicted state and input trajectories are given by $\mathbf{x}_{\mathrm{t}}(\cdot) = [\mathbf{x}_{\mathrm{0,t}}\,, \mathbf{x}_{\mathrm{1,t}}\,, \ldots\,, \mathbf{x}_{\mathrm{N,t}}]$ and $\mathbf{u}_{\mathrm{t}}(\cdot) = [\mathbf{u}_{\mathrm{0,t}}, \mathbf{u}_{\mathrm{1,t}}\,, \ldots\,, \mathbf{u}_{\mathrm{N-1,t}}]$. The term $\mathbf{f}^{\mathrm{dt}}(\mathbf{x}_{\mathrm{k,t}},\mathbf{u}_{\mathrm{k,t}})$ denotes the nonlinear discrete-time system dynamics presented in Section 5.2.1. Equations (8e) and (8f) reflect the preference for small slip angles on the vehicle's performance, where $\alpha_{\mathrm{f,sat}}$ and $\alpha_{\mathrm{r,sat}}$ are equal to 6°. Equation (8d) corresponds to the steering angle saturation limit imposed by the steering actuator of the RC vehicle, whose value is $u_{\mathrm{sat}} = 20°$. Furthermore, the cost function terms Φ and L are given by

$$\Phi(\mathbf{x}_{\mathrm{N,t}}) = \tilde{\mathbf{x}}_{\mathrm{N,t}}^{\mathrm{T}} P \tilde{\mathbf{x}}_{\mathrm{N,t}}\,, \tag{9}$$

$$L(\mathbf{x}_{\mathrm{k,t}},\mathbf{u}_{\mathrm{k,t}}) = \tilde{\mathbf{x}}_{\mathrm{k,t}}^{\mathrm{T}} Q \tilde{\mathbf{x}}_{\mathrm{k,t}} + \mathbf{u}_{\mathrm{k,t}}^{\mathrm{T}} R \mathbf{u}_{\mathrm{k,t}}\,, \tag{10}$$

where $\tilde{\mathbf{x}}_{\mathrm{k,t}}$ represents the difference between the reference $\mathbf{x}_{\mathrm{ref,k,t}}$ and the predicted $\mathbf{x}_{\mathrm{k,t}}$ states.

At each time step t, the input $\mathbf{u}_{\mathrm{NMPC}}(t) = \mathbf{u}_{\mathrm{0,t}}^*$ is applied to the system during the time interval $[t; t+\delta]$ with δ being the sampling time and $\mathbf{u}_{\mathrm{0,t}}^*$ the first term of the optimal control input sequence. To achieve a real-time NMPC implementation, each NMPC iteration needs to be completed within the sampling time.

The reference state vector $\mathbf{x}_{\mathrm{ref,k,t}}$ and the penalty function term $PF_{\mathrm{goal,k,t}}$ are computed by the global path generator in Section 3.1.1. On the contrary, $PF_{\mathrm{obs,k,t}}$ is associated with the local path generator of Section 3.1.2.

3.1.1. Global Path Planner

The global path planner comprises the procedures that drive the RC vehicle towards the goal position, which in this thesis corresponds to the computation of the reference trajectory and the usage of the penalty function PF_{goal}.

First, the reference trajectory computation is performed offline and consists of the shortest straight line connecting the initial (X_0,Y_0) and final $(X_{\mathrm{f}},Y_{\mathrm{f}})$ coordinate points. Then, at each time step t, the reference trajectory $\mathbf{x}_{\mathrm{ref}}(\cdot)$ is updated. Specifying the reference trajectory as a function of time t has no advantage since it would lead to error accumulation, especially when new obstacles are detected and the vehicle needs to deviate from them [22]. Therefore, at each time step, the reference trajectory is composed by the set of coordinate points on the shortest straight line with the same X or Y coordinate as the predicted X and Y states of the RC vehicle, depending on which one is nearer the final $(X_{\mathrm{f}},Y_{\mathrm{f}})$ position.

The penalty function term $PF_{\mathrm{goal,k,t}}$ behaves as an attractive field that guides the RC vehicle towards the destination point at every time step k within the look-ahead horizon N. Without this term, the RC vehicle would, most likely, excessively deviate from the obstacles and stop planning its movement with respect to the goal position. The potential-like function $PF_{\mathrm{goal,k,t}}$ is given by

$$PF_{\mathrm{goal,k,t}} = \frac{1}{2} K_{\mathrm{goal}} d_{\mathrm{goal,k,t}}^2\,, \tag{11}$$

where K_{goal} corresponds to the weighting parameter and $d_{\mathrm{goal,k,t}}$ is the distance, at time t, between the predicted $(X_{\mathrm{k,t}},Y_{\mathrm{k,t}})$ and the goal coordinates $(X_{\mathrm{f}},\ Y_{\mathrm{f}})$.

3.1.2. Local Path Planner

This thesis implements a local path planner that uses repulsion forces to drive the RC vehicle away from obstacles. The local path planner has access to the most updated LIDAR data available in order to react to changes in the environment and allow the RC vehicle to drive in unknown scenarios.

Weighted Distance Method

The weighted distance (WD) method uses the distance to the nearest obstacle to construct the repulsive potential field from the obstacles. Therefore, the term $PF_{\mathrm{obs,k,t}}$ is given by

$$PF_{\mathrm{obs,k,t}} = K_{\mathrm{obs}} \frac{v}{\min_{\mathrm{i}}(d_{\mathrm{obs,k,t,i}}) + \epsilon}\,, \tag{12}$$

where K_{obs} represents the weighting parameter and ϵ a small positive constant to avoid singularities when the distance to the nearest obstacle $\min_{\mathrm{i}}(d_{\mathrm{obs,k,t,i}})$ is null.

In this method, the vehicle's dimension is indirectly considered, which is a positive characteristic. However, its actual moving direction is not taken into account. For this reason, equally distant obstacles represent the same level of threat to the RC vehicle, regardless of its moving direction (Fig.1). This fact provides a major drawback of distance-based methods.

Modified Parallax Method

The modified parallax (MP) method, as humans, perceives the distance to an object by the angle between two different lines of sight (parallax principle). However, the MP method modifies the parallax principle in order to reflect the actual moving direction of the vehicle as illustrated in Figure 2.

Suppose at time t, the set of obstacle points detected on the forward and side sensing ranges are given by p and q. The coordinates of the points belonging to the set p in the vehicle body

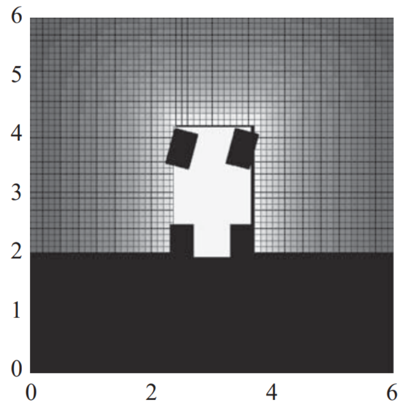

Figure 1: Penalty function of WD method. Connected points of the same colour represent the same level of threat. Points of lighter colour represent a higher level of threat [21].

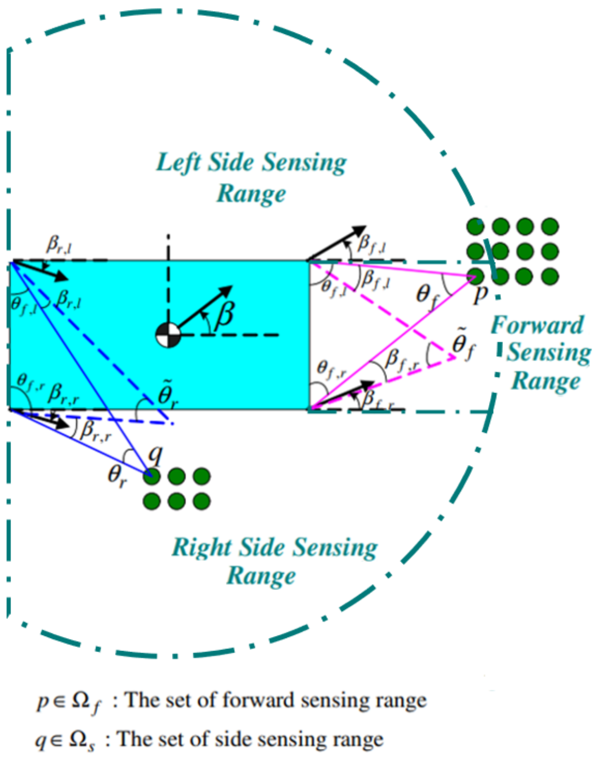

Figure 2: Modified parallax principle (cf.[21]).

frame throughout the prediction horizon N, $(p_{\mathrm{x,i,k,t}}, p_{\mathrm{y,i,k,t}})$, are

$$p_{\mathrm{x,i,k,t}} = (p_{\mathrm{X,i,t}} - X_{\mathrm{k,t}})\cos\varphi_{\mathrm{k,t}} + (p_{\mathrm{Y,i,t}} - Y_{\mathrm{k,t}})\sin\varphi_{\mathrm{k,t}}\,,$$
$$p_{\mathrm{y,i,k,t}} = (-p_{\mathrm{X,i,t}} + X_{\mathrm{k,t}})\sin\varphi_{\mathrm{k,t}} + (p_{\mathrm{Y,i,t}} - Y_{\mathrm{k,t}})\cos\varphi_{\mathrm{k,t}}\,,$$

where i represents the i-th point on the set p and $(X_{\mathrm{k,t}}, Y_{\mathrm{k,t}})$ are the predicted inertial coordinates of the RC vehicle, at time t, with $k = 0, \ldots, N-1$. Furthermore, $(p_{\mathrm{X,i,t}}, p_{\mathrm{Y,i,t}})$ represents the inertial coordinates of the i-th obstacle in the forward sensing range measured at time t. The coordinates of the points belonging to the set q in the vehicle body frame, $(q_{\mathrm{x,i,t}}, q_{\mathrm{y,i,t}})$, follow the same logic as the set p.

The expressions to calculate the parallax angles, $\theta_{\mathrm{fl,i,k,t}}$, $\theta_{\mathrm{fr,i,k,t}}$, $\theta_{\mathrm{rl,i,k,t}}$ and $\theta_{\mathrm{rr,i,k,t}}$, are

$$\theta_{\mathrm{fl,i,k,t}} = \tan^{-1}\left(\frac{p_{\mathrm{x,i,k,t}} - \frac{L}{2}}{\frac{W}{2} - p_{\mathrm{y,i,k,t}}}\right), \quad \theta_{\mathrm{fr,i,k,t}} = \tan^{-1}\left(\frac{p_{\mathrm{x,i,k,t}} - \frac{L}{2}}{\frac{W}{2} + p_{\mathrm{y,i,k,t}}}\right),$$

$$\theta_{\mathrm{rl,i,k,t}} = \tan^{-1}\left(\frac{q_{\mathrm{x,i,k,t}} + \frac{L}{2}}{\frac{W}{2} - q_{\mathrm{y,i,k,t}}}\right), \quad \theta_{\mathrm{rr,i,k,t}} = \tan^{-1}\left(\frac{q_{\mathrm{x,i,k,t}} + \frac{L}{2}}{\frac{W}{2} + q_{\mathrm{y,i,k,t}}}\right).$$

The moving direction of the four vertices of the vehicle are denoted as $\beta_{\mathrm{fl,k,t}}$, $\beta_{\mathrm{fr,k,t}}$, $\beta_{\mathrm{rl,k,t}}$, $\beta_{\mathrm{rr,k,t}}$ and are given by

$$\beta_{\mathrm{fl,k,t}} = \tan^{-1}\left(\frac{v\sin\beta_{\mathrm{k,t}} + \frac{L}{2}\dot{\varphi}_{\mathrm{k,t}}}{v\cos\beta_{\mathrm{k,t}} - \frac{W}{2}\dot{\varphi}_{\mathrm{k,t}}}\right), \tag{13}$$

$$\beta_{\mathrm{fr,k,t}} = \tan^{-1}\left(\frac{v\sin\beta_{\mathrm{k,t}} + \frac{L}{2}\dot{\varphi}_{\mathrm{k,t}}}{v\cos\beta_{\mathrm{k,t}} + \frac{W}{2}\dot{\varphi}_{\mathrm{k,t}}}\right), \tag{14}$$

$$\beta_{\mathrm{rl,k,t}} = \tan^{-1}\left(\frac{v\sin\beta_{\mathrm{k,t}} - \frac{L}{2}\dot{\varphi}_{\mathrm{k,t}}}{v\cos\beta_{\mathrm{k,t}} - \frac{W}{2}\dot{\varphi}_{\mathrm{k,t}}}\right), \tag{15}$$

$$\beta_{\mathrm{rr,k,t}} = \tan^{-1}\left(\frac{v\sin\beta_{\mathrm{k,t}} - \frac{L}{2}\dot{\varphi}_{\mathrm{k,t}}}{v\cos\beta_{\mathrm{k,t}} + \frac{W}{2}\dot{\varphi}_{\mathrm{k,t}}}\right). \tag{16}$$

The Algorithm 1 describes the steps to compute the front and rear MP angles for every point in the sets p and q. Note that n_{p} and n_{q} correspond to the total number of points in each set p and q.

Algorithm 1 Front and rear MP angles algorithm.

```
 1: for $i = 1, \ldots, n_{\mathrm{p}}$ do
 2:     if $0 < (\theta_{\mathrm{fl,i,k,t}} - \beta_{\mathrm{fl,k,t}}) + (\theta_{\mathrm{fr,i,k,t}} + \beta_{\mathrm{fr,k,t}}) < \pi$ then
 3:         $\tilde{\theta}_{\mathrm{f,i,k,t}} = \pi - [(\theta_{\mathrm{fl,i,k,t}} - \beta_{\mathrm{fl,k,t}}) + (\theta_{\mathrm{fr,i,k,t}} + \beta_{\mathrm{fr,k,t}})]$
 4:     else
 5:         $\tilde{\theta}_{\mathrm{f,i,k,t}} = 0$
 6:     end if
 7: end for
 8: for $i = 1, \ldots, n_{\mathrm{q}}$ do
 9:     if $0 < (\theta_{\mathrm{rl,i,k,t}} - \beta_{\mathrm{rl,k,t}}) + (\theta_{\mathrm{rr,i,k,t}} + \beta_{\mathrm{rr,k,t}}) < \pi$ then
10:         $\tilde{\theta}_{\mathrm{r,i,k,t}} = \pi - [(\theta_{\mathrm{rl,i,k,t}} - \beta_{\mathrm{rl,k,t}}) + (\theta_{\mathrm{rr,i,k,t}} + \beta_{\mathrm{rr,k,t}})]$
11:     else
12:         $\tilde{\theta}_{\mathrm{r,i,k,t}} = 0$
13:     end if
14: end for
```

Let the points of the obstacles with the largest front and rear MP angles be denoted by $p_{\mathrm{i,max}}$ and $q_{\mathrm{i,max}}$, then the potential-like function $PF_{\mathrm{obs,k,t}}$ at step k is given by

$$PF_{\mathrm{obs,k,t}} = \begin{cases} 0 & \text{if no obstacles}\,, \\ K_{\mathrm{obs}}\exp\left(\frac{\tilde{\theta}_{\mathrm{f,k,t}}}{\theta_{\mathrm{cf}}(v)}\right) & \text{if } p_{\mathrm{i,max}}\,, \\ K_{\mathrm{obs}}\exp\left(\frac{\tilde{\theta}_{\mathrm{r,k,t}}}{\theta_{\mathrm{cr}}(v)}\right) & \text{if } q_{\mathrm{i,max}}\,, \\ K_{\mathrm{obs}}\exp\left(\frac{\tilde{\theta}_{\mathrm{f,k,t}}}{\theta_{\mathrm{cf}}(v)} + \frac{\tilde{\theta}_{\mathrm{r,k,t}}}{\theta_{\mathrm{cr}}(v)}\right) & \text{if } p_{\mathrm{i,max}} \text{ and } q_{\mathrm{i,max}}\,, \end{cases} \tag{17}$$

where the front and rear MP angles, $\tilde{\theta}_{\mathrm{f,k,t}}$ and $\tilde{\theta}_{\mathrm{r,k,t}}$ are computed for the respective $p_{\mathrm{i,max}}$ and $q_{\mathrm{i,max}}$ points. The terms

$\theta_{\mathrm{cf}}(v)$ and $\theta_{\mathrm{cr}}(v)$ correspond to the critical MP values that depend on the vehicle velocity v. These values should decrease as the speed increases and, therefore, they are defined as

$$\theta_{\mathrm{cf}}(v) = \frac{K_{\mathrm{cf}}}{v}\,, \qquad \theta_{\mathrm{cr}}(v) = \frac{K_{\mathrm{cr}}}{v}\,,$$

where K_{cf} and K_{cr} represent weighting parameters.

The MP method not only considers the vehicle dimensions in an explicit manner as shown in (16) but also the moving direction of the vehicle. For this reason, in the MP method, obstacles at the same distance to the vehicle do not necessarily represent the same level of threat, which is a clear advantage over distance-based methods (Fig. 3).

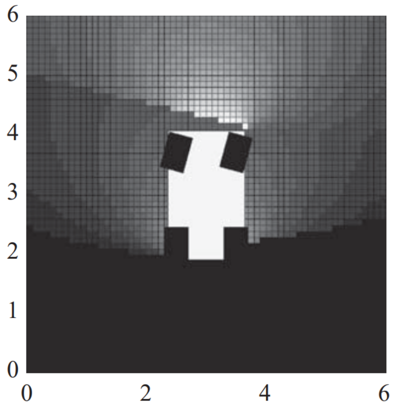

Figure 3: Penalty function of MP method. Connected points of the same colour represent the same level of threat to the vehicle. Points of lighter colour represent a higher level of threat [17].

3.2. Solution Methods for NLPs in NMPC

NMPC is based on the real-time optimization of an NLP problem which is still computationally challenging. In NMPC, suboptimal solutions that are computed fast enough are preferred over precise solutions that require more computational effort [19]. The NMPC problem with the gradient descent method [20] as NLP solver can lead to a low enough computational load for real-time control applications [22]. Therefore, this method is used to solve the NLP problem in (8) and, in Section 4.3, its performance is compared with the state-of-the-art IPOPT solver.

3.2.1. Gradient Descent Method

The gradient descent method is an efficient NLP solver that uses the indirect method of Lagrange multipliers to perform the optimization process [20]. In this method, the cost function is augmented with the equality and inequality constraints vectors as follows

$$\mathbf{F}_{\mathrm{aug}}(\mathbf{w},\boldsymbol{\lambda},\boldsymbol{\mu}) = \Phi(\tilde{\mathbf{x}}_{\mathrm{N,t}}) \sum_{k=0}^{N-1} + L(\tilde{\mathbf{x}}_{\mathrm{k,t}},\mathbf{u}_{\mathrm{k,t}}) + \boldsymbol{\lambda}_{\mathrm{k+1}}^{\mathrm{T}}\mathbf{G}(\mathbf{x}_{\mathrm{k,t}},\mathbf{u}_{\mathrm{k,t}}) + \phi(\mathbf{x}_{\mathrm{k,t}},\mathbf{u}_{\mathrm{k,t}}) + PF_{\mathrm{goal,k,t}} + PF_{\mathrm{obs,k,t}}\,, \tag{18}$$

where $\mathbf{w}$ corresponds to the vector of decision variables, *i.e.*, $\mathbf{w} = [\mathbf{x}_0, ..., \mathbf{x}_{\mathrm{N}}, \mathbf{u}_0, ..., \mathbf{u}_{\mathrm{N-1}}]$, $\boldsymbol{\lambda}_{\mathrm{k}} \in \mathbb{R}^{n_{\mathrm{x}}}$ denotes the Lagrange multiplier vector, and $\mathbf{G}$ is the vector of equality constraints in (8b) and (8c). Similarly to the approach used to solve the path planning problem, this thesis uses a penalty function ϕ to integrate the inequality constraints in (8d)-(8f) into $\mathbf{F}_{\mathrm{aug}}$. After careful analysis of simulation results, the selected penalty function ϕ consists of a logarithmic barrier function that is given by

$$\boldsymbol{\phi}(\mathbf{w}) = \sum_{i=1}^{n_{\mathrm{h}}} \boldsymbol{\tau}_{\mathrm{i}}\,\boldsymbol{\phi}_{\mathrm{i}}(\mathbf{w})\,, \qquad \boldsymbol{\phi}_{\mathrm{i}}(\mathbf{w}) = -\log(-\mathbf{H}_{\mathrm{i}}(\mathbf{w}))\,, \tag{19}$$

where $i \in \{1, \ldots, n_{\mathrm{h}}\}$, $\boldsymbol{\tau}_{\mathrm{i}} \in \mathbb{R}$ is the barrier parameter and $\mathbf{H} \in \mathbb{R}^{n_{\mathrm{h}}}$ corresponds to the vector of inequality constrains in (8d)-(8f). The logarithmic barrier function adds a penalty that tends to infinity as $\mathbf{w}$ approaches the inequality constrained with equality. This method provides an alternative to the exterior approaches especially when feasibility has to be strictly fulfilled.

All things considered, a constraint optimization problem is transformed into an unconstrained one. Consequently, the gradient descent method minimizes, simultaneously, the objective function and the infeasibility of the constraints. The gradient descent numerical algorithm is described in [20].

4. Comparative Studies

This section presents two comparative studies: the first one examines the performance of the two local path planners, WD and MP, for a cluttered scenario. The second study compares the gradient descent method and the IPOPT solver in terms of computational efficiency and feasibility of solutions.

4.1. General Simulation Setup

The simulations were performed in Matlab 2019 [14] and CasADI [3] on the desktop Intel(R) Core(TM) I5-5200U CPU @ 2.20 GHz. The IPOPT software package used is included in CasADI installations. The NMPC controller and gradient descent method were first formulated using CasADI symbolic syntax and Matlab programming language. The developed code was converted into C-code using CasADI's C-code generation tool and, afterwards, compiled into a MEX function in order to be executed from Matlab.

4.2. Local Path Planners Performance

The cluttered scenario is composed of fourteen 2m size squared obstacles and two circular obstacles with 1m of radius. The displacement between obstacles is 4 m in both X and Y directions. In this study, the RC vehicle navigates at a constant speed v of 1.5 m/s and the NMPC scheme uses $N = 20$ and $\delta = 0.1$ s.

The WD method is not capable to avoid all obstacles in a cluttered scenario. On the contrary, the MP performs successfully the obstacle avoidance task (Fig. 4). Furthermore, in contrast to the MP, the NMPC with the WD method violates the input constraints limits in $t = 8.2$ s and $t = 14.5$ s (Fig. 5).

The average NMPC computational times of the WD and MP methods in the cluttered scenario are 0.035 s and 0.173 s. Therefore, on average, both methods fulfill the time constraint of $\delta = 0.1$ s. However, the maximum run-times of the WD and MP methods are 0.1087 s and 0.0291 s, meaning that the WD occasionally violates the NMPC computational requirements.

In conclusion, the MP method outperformed the WD approach in terms of run-time efficiency and computation of collision-free trajectories in the cluttered scenario.

4.3. NLP Solvers Performance

The scenario is composed of 9 blocks of 2 m x 2 m size separated by 6.25 m and 5 m in X and Y directions. In this simulation, the RC vehicle moves at a constant speed v of 1.75 m/s and the NMPC scheme uses $N = 40$ and $\delta = 0.05$ s.

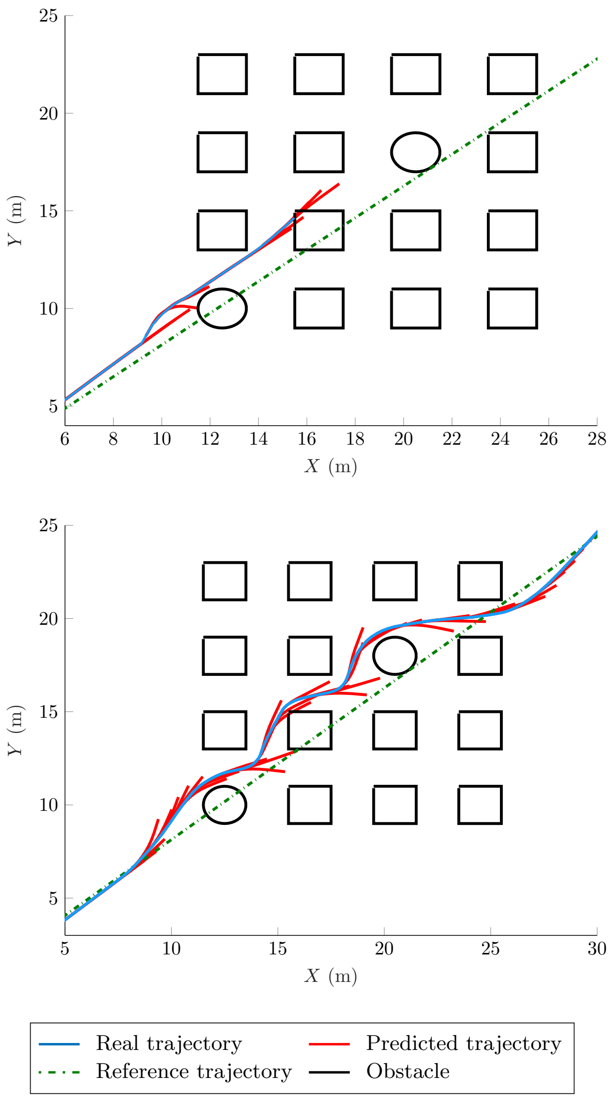

Figure 4: Computed XY trajectories by WD (top) and MP (down) methods in the cluttered scenario.

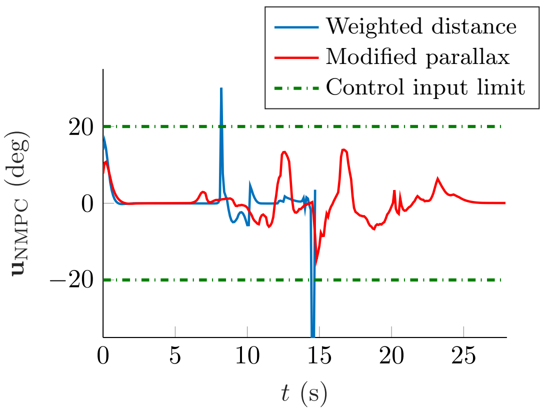

Figure 5: Computed control input sequence $\mathbf{u}_{\text{NMPC}}(\cdot)$ by WD and MP methods in the cluttered scenario.

Both NLP solvers present identical collision-free trajectories (Fig.6) and kept the control input sequence within the saturation limits (Fig.7). However, the control input solution obtained by the gradient descent method is less oscillatory.

The average and maximum NMPC computational times with the IPOPT are 44.6 ms and 72.2 ms, while for the gradient search are 1.6 ms and 17.5 ms. Contrarily to the gradient descent method, not all computations of the IPOPT method were completed within the time constraint of $\delta = 50\,\text{ms}$. A possible reason for this occurrence is that IPOPT is a complex interior-point algorithm capable of solving general nonlinear optimization problems, while the gradient descent method is a simple algorithm that can be tuned for this specific NLP problem, for example, by changing the step-size and barrier parameter. Nevertheless, on average, the IPOPT satisfied the computational requirements, which turned out to be enough to compute a collision-free trajectory.

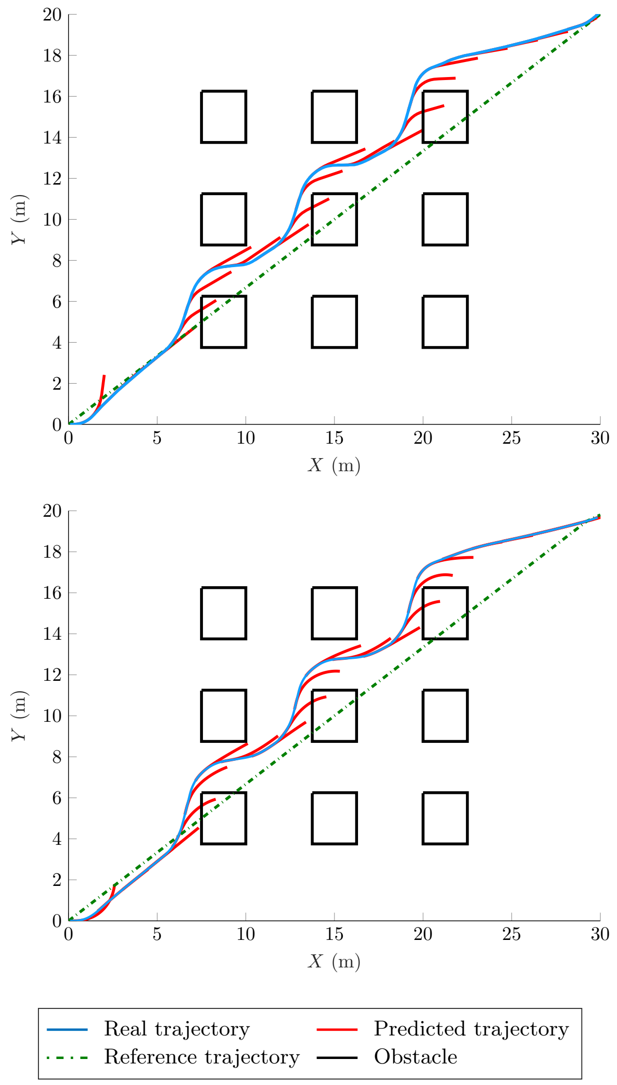

Figure 6: Computed XY trajectories by IPOPT (top) and gradient descent method (down).

5. Implementation

This chapter presents the implementation steps of the purposed NMPC scheme on the RC vehicle.

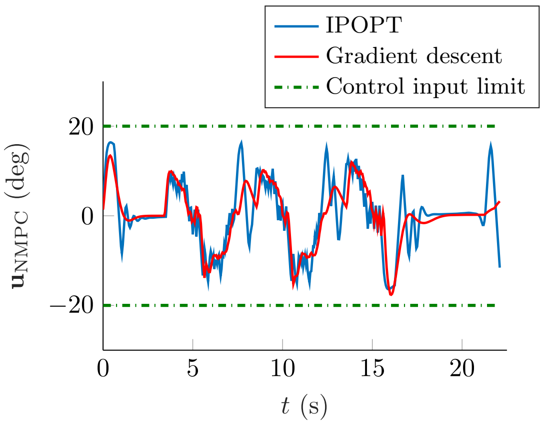

Figure 7: Computed control input sequence $\mathbf{u}_{\text{NMPC}}(\cdot)$ by IPOPT and gradient descent method.

5.1. Software Tools and Code Generation Procedure

To rapidly develop and test the performance of the NMPC framework and gradient descent NLP solver with the logarithmic barrier as penalty function, Matlab 2019 [14] and CasADI [3] were used on the Intel(R) Core(TM) I5-5200U CPU @ 2.20 GHz desktop. Afterwards, the Matlab code was converted into C-code using the CasADI's C-code generation tool. Finally, the generated C code was integrated into the Python software package that controls the RC vehicle using the SWIG [2] software tool.

5.2. Integration Methods in the NMPC context

NMPC relies on the model (1) to predict the future evolution of a system. To solve this system of ODEs on a computational unit, first, it is necessary to discretize the continuous-time equations using a numerical integration method. The goal is to choose an adequate integration method with a suitable integration step size Δt that provides reliable predictions of the system's evolution and allows the NMPC algorithm to meet the computational constraints.

5.2.1. Integration Method Validation

Three numerical integration methods are tested for the step sizes $\Delta t \in [0.01, 0.1]$ (s). Among them are the Euler method, Fourth Order Runge-Kutta (RK4) method with one integration step, $M = 1$, named RK4$_{\text{M=1}}$, and a RK4 method provided by CasADI that considers 25 intermediate steps, RK4$_{\text{M=25}}$.

The advanced and variable step-size numerical integrator CVodes [1] is used to evaluate the performance of the before mentioned integration methods, whose accuracy is accessed by

$$d_{\text{i}} = \sqrt{X^2_{\text{cvodes,i}} + Y^2_{\text{cvodes,i}}}\,, \qquad \hat{d}_{\star,\text{i}} = \sqrt{X^2_{\star,\text{i}} + Y^2_{\star,\text{i}}}\,, \tag{20}$$

$$RMSE = \sqrt{\sum_{i=0}^{\mathcal{N}} \frac{(\hat{d}_{\star,\text{i}} - d_{\text{i}})^2}{\mathcal{N}+1}}\,, \,(m) \tag{21}$$

where $X_{\text{cvodes,i}}$ and $Y_{\text{cvodes,i}}$ represent the X and Y states computed by the CVodes integration method for the step i in the simulation horizon $\mathcal{N}$. The subscript $\star$ corresponds to either Euler, RK4$_{\text{M=1}}$ or RK4$_{\text{M=25}}$ methods. Furthermore, d_{i} and $\hat{d}_{\star,\text{i}}$ denote, respectively, the distance to the origin of the referential computed at step i by CVodes and the Euler, RK4$_{\text{M=1}}$, and RK4$_{\text{M=25}}$ methods.

In general, for high integration step sizes, both RK4 methods lead to a much better performance than the Euler method (Fig.8). Moreover, from the observation of Table 3, one can conclude that, with the RK4$_{\text{M=25}}$ integration method, 25 intermediate steps were used without a considerable increase in the computational time. Additionally, it can be verified that the RK4$_{\text{M=25}}$ method presents a low RMSE value, which means that its XY state trajectory approximates accurately the CVodes' XY results. Thus, the RK4$_{\text{M=25}}$ with $\Delta t = 0.1$ s is selected as the integration scheme in the experimental phase.

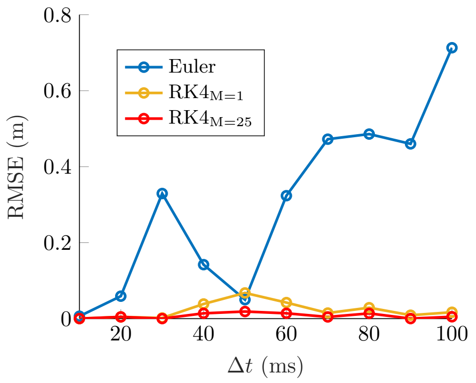

Figure 8: RMSE, in meters, of Euler, RK4$_{\text{M=1}}$, and RK4$_{\text{M=25}}$ for $\Delta t \in [0.01, 0.1]$ (s).

Parameter	CVodes	RK4$_{\text{M=1}}$	RK4$_{\text{M=25}}$
Avg. run-time	8.2 ms	1.6 ms	1.9 ms
RMSE	-	0.0168 m	0.0049 m

Table 3: Average computational time, in milliseconds, of CVodes, RK4$_{\text{M=1}}$, RK4$_{\text{M=25}}$, and correspondent RMSE values, in meters, for $\Delta t = 0.1$ s.

5.3. Sampling Time and Computational Delay of NMPC

The sampling time determines the resolution of the predictions and how frequent control actions can be applied to the system [4]. The integration step size Δt and the sampling time δ are related by the following inequality $\Delta t \leq \delta$, which, according to Section 5.2.1, leads to $\delta \geq 0.1$ s. Moreover, real systems are affected by computational delays which provide a new lower-bound to the sampling time δ. Thus, the sampling time δ value of 0.2 s is used in the NMPC implementation, meaning that, for each sampling time δ, the RK4$_{\text{M=25}}$ performs $\delta/\Delta t = 2$ integration steps.

To solve the computational delay issue on the RC vehicle, this thesis implements the delay compensation by prediction. This approach is often used in practical NMPC frameworks with significant computational time [8]. The delay compensation by prediction consists of simulating the state at which the system will be when the NLP problem is solved. This can be performed using the NMPC system model and the open-loop control inputs that will be applied to the system in the meantime [8]. In Figure 9 the delay compensation by prediction is schematized for the implemented NMPC scheme with $\delta = 0.2$ s and $\Delta t = 0.1$ s. The NMPC computations that start at time $t + \delta$ are supposed to finish at time $t + 2\delta$. For this reason, at time $t+\delta$, the NMPC problem optimizes for the state that is δ

seconds ahead, *i.e.*, $t+2\delta$. The prediction of such state is done using the RC vehicle model and the control input $\mathbf{u}_{\mathrm{NMPC}}(\cdot)$ obtained in the previous optimization period $[t,t+\delta]$. This process is repeated throughout the time.

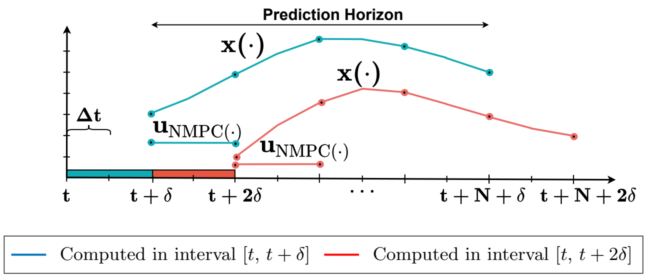

Figure 9: Delay compensation by prediction for NMPC with $\delta = 0.2\,\mathrm{s}$ and $\Delta t = 0.1\,\mathrm{s}$.

6. Experimental Validation

The NMPC algorithm formulated in Chapter 3 is validated using three different experiments: A, B, and C. The NMPC problem is solved by the gradient descent method with the logarithmic barrier as penalty function. In order to perform obstacle avoidance, the NMPC uses the modified parallax method. Moreover, the selected integration method is the $\mathrm{RK4}_{\mathrm{M}=25}$ with $\Delta t = 0.1\,\mathrm{s}$ and the NMPC sampling time is $\delta = 0.2\,\mathrm{s}$. Finally, the NMPC algorithm is based on the delay compensation by prediction approach to solve the computational delay issue. The driving scenario corresponds to the interior of the institute's building and the desired destination position is $(X_{\mathrm{f}},Y_{\mathrm{f}})$=(30,35) m.

According to the comparative studies, the computation of the reference trajectory $\mathbf{x}_{\mathrm{ref}}$ as described in Section 3.1.1 is an adequate approach, however, in the experimental phase that is not verified. When the reference trajectory is considered, the NMPC controller is difficult to tune and the RC vehicle collides often with the walls of corridor. This is because a straight line connecting the initial and goal positions is too far off from the desired trajectory, especially from $X = 5\,\mathrm{m}$ to $X = 15\,\mathrm{m}$. For this reason, the matrices Q and P were set to $\mathrm{diag}(0,0,0,0,0)$, which led to more freedom of movement and successful experiments.

6.1. Experiment A

Experiment A tests the performance of the NMPC controller for a low vehicle velocity, $v = 0.6\,\mathrm{m/s}$, and short prediction horizon, $N = 5$, which naturally results in a short prediction time, $T = 1\,\mathrm{s}$.

The RC vehicle with the purposed NMPC controller successfully avoids the walls of the institute until the destination point (Fig. 10). The short prediction time T and the low vehicle velocity v resulted in predicted XY trajectories with 0.6 m of length. Consequently, the NMPC only detects a potential collision on its horizon when the RC vehicle is already too close to the obstacles. For this reason, the vehicle is forced to avoid the obstacles too aggressively, which can be observed from $(23,16)$ [m] to $(23,18)$ [m]. Moreover, the RC vehicle presents an oscillatory movement from $(7.5,6.3)$ [m] to $(12.2,6.2)$ [m]. This is because the RC vehicle is strongly attracted by the goal position and therefore tries to drive in its direction. Then, the vehicle changes its direction of movement because of the left lateral wall. Afterwards, the vehicle detects the right lateral walls and tries to avoid it by steering to the left. This process is repeated in time, which culminates in its characteristic undulated movement. Comparing the real and desired final positions, the obtained errors of the state variables X and Y are relatively small, *i.e.*, 4 cm and 8 cm, respectively.

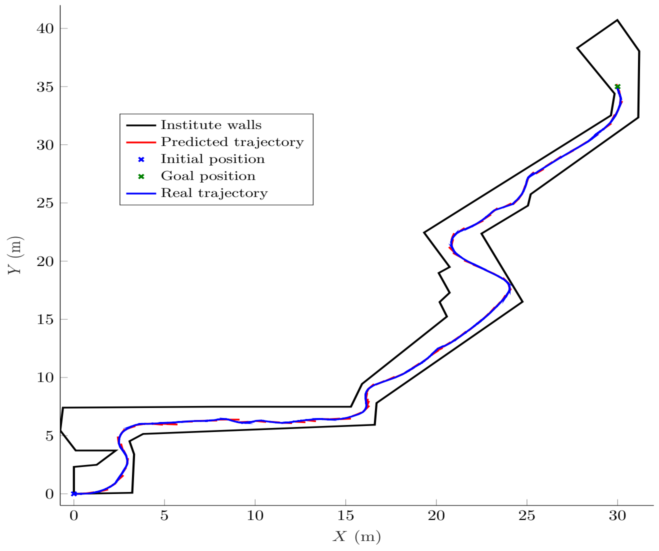

Figure 10: Real and predicted XY trajectories of RC vehicle in experiment A.

The purposed controller computes a control input sequence within the saturation limits (Fig. 11). However, the control input sequence given to the RC vehicle presents several oscillations in short periods of time, for instance, in the time interval $t \in [11,18]$ (s). Finally, the average and maximum computational times of the NMPC algorithm are 0.1025 s and 0.1980 s. Therefore, the computational requirements are satisfied in all time steps since the NMPC iterations finish before the sampling time $\delta = 0.2\,\mathrm{s}$.

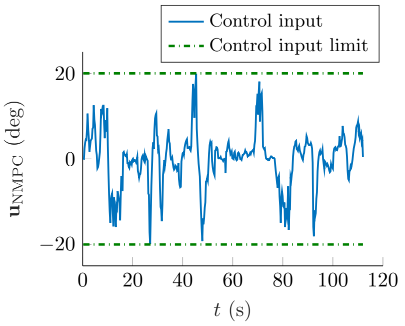

Figure 11: Control input $\mathbf{u}_{\mathrm{NMPC}}(\cdot)$ in experiment A.

6.2. Experiment B

Experiment B tests the NMPC controller performance for a low vehicle velocity, $v = 0.6\,\mathrm{m/s}$, and twice the prediction horizon of experiment A, $N = 10$. This naturally results in a longer prediction time T of $= 2\,\mathrm{s}$.

The purposed NMPC controller successfully avoids the walls of the institute until the goal position (Fig. 12). Comparing the real and desired final positions, the errors are 2.7 cm on the X-axis and 4.5 cm on the Y-axis.

Moreover, the NMPC controller computes a control input sequence within the saturation limits for all time steps (Fig.

13). The average and maximum computational times of the NMPC algorithm are 0.1478 s and 0.2305 s, meaning that, occasionally, the purposed NMPC controller does not satisfy the computational requirement of $\delta = 0.2$ s.

Comparing the results of experiments A and B, the experiment B presents slightly less oscillations in the corridor from $(3.8, 5.1)$ m to $(15.3, 7.5)$ m. Moreover, in experiment B, the RC vehicle describes the curve from $(15.3, 7.5)$ m to $(17, 8)$ m and avoids the front wall, from $(25, 2.5)$ m to $(25.3, 25.8)$ m, more smoothly. Both experiments performed an aggressive steering manoeuvre to successfully avoid the front wall from $(24.7, 16.5)$ m to $(22.5, 22.4)$ m. Furthermore, experiments A and B approximate reasonably the goal position. However, the first experiment presents a final error that is 1.5 times higher than in experiment B. As expected, by using a longer prediction horizon N in experiment B than in experiment A, the computational time of the NMPC algorithm increased. On average, the growth was 44.17 %.

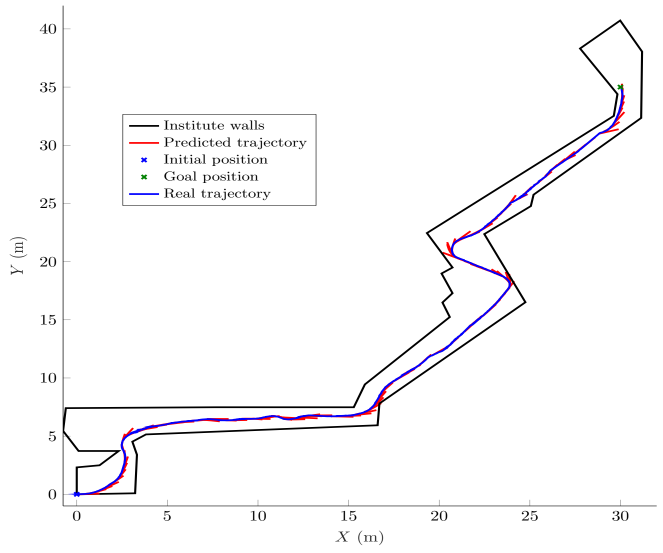

Figure 12: Real and predicted XY trajectories of RC vehicle in experiment B.

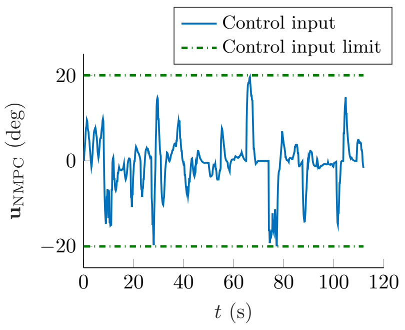

Figure 13: Control input $\mathbf{u}_{\text{NMPC}}(\cdot)$ in experiment B.

6.3. Experiment C

Experiment C tests the NMPC controller performance for a higher vehicle velocity than in experiments A and B, $v = 0.9$ m/s. The NMPC prediction horizon N is equal to the one used in experiment B, *i.e.*, $N = 10$.

The RC vehicle with the purposed NMPC controller successfully avoids the walls of the institute until the destination point (Fig. 14). In experiment C, the NMPC algorithm predicts XY trajectories with 1.8 m of length. Therefore, the NMPC algorithm detects a potential collision in its horizon when the RC vehicle is still at a considerable safe distance from the obstacles. For instance, from $(22, 14)$ m to $(22.5, 22.4)$ m, the RC vehicle turns left at a bigger margin from the front wall than in experiments A and B. Furthermore, in the experiment C, the RC vehicle presents undulated movement in the corridors, similarly to what was verified in the first two experiments. Comparing the real and desired final positions, the errors are 4 cm on the X-axis and 7 cm on the Y-axis.

The NMPC controller computes a control sequence that is within the saturation limits for all time steps (Fig. 15). The average and maximum computational times of the NMPC algorithm are 0.1678 s and 0.2507 s. Thus, the computational requirement of $\delta = 0.2$ s was not fulfilled for all time steps. Nevertheless, this didn't represent a problem since, on average, the computational time is still far from the limit δ.

Comparing the results of experiments B and C, the last one presents, in general, a bigger margin from the obstacles, due to the larger length of the predicted XY states. Moreover, experiments B and C present a soft real-time NMPC implementation since the time requirements are not fulfilled in all times. As the velocity increases, the optimization problem becomes more computationally demanding for the NLP solvers. Therefore, from experiments B to C, the computational time increased. On average, it was verified a computation growth of 13.56 %.

The different NMPC setups present the same shortcomings when driving the RC vehicle through the experimental scenario, which are exiting the laboratory room in (3,5) m, oscillatory movement in the long horizontal corridor after the laboratory room, agressive maneuver to avoid the large front wall from $(25, 2.5)$ m to $(25.3, 25.8)$ m, and entering in the new corridor at $(22.5, 21)$ m, which is a similar problem as exiting the laboratory room. It is believed that the online computation of a reference trajectory, instead of the offline approach designed in Section 3.1.1, would solve or, at least, attenuate the before mentioned problems.

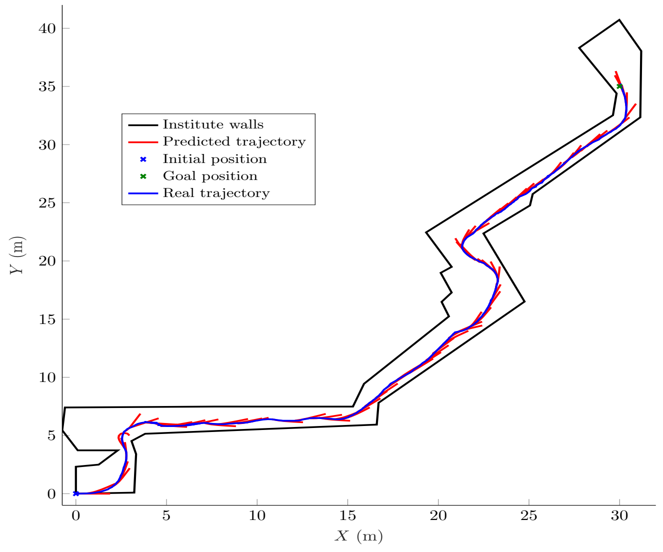

Figure 14: Real and predicted XY trajectories of RC vehicle in experiment C.

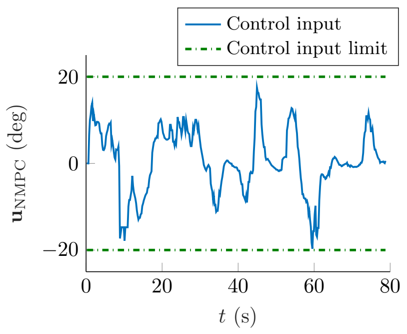

Figure 15: Control input $\mathbf{u}_{\text{NMPC}}(\cdot)$ in experiment C.

7. Conclusions

Based on the simulation results, the current thesis purposes an NMPC algorithm that uses the MP as local path planner, the gradient descent method as NLP solver, and the logarithmic barrier as penalty function. Moreover, during the implementation of such controller on the RC vehicle, a significant delay between the state information and the control input applied to the system was verified. To solve this issue, the delay compensation by prediction was implemented.

Experimental tests were performed on an RC vehicle to prove the effectiveness of the designed NMPC controller. For a low prediction horizon and vehicle velocity, the RC vehicle was able to navigate through an unknown scenario without any collisions with obstacles. As expected, the increase of the prediction horizon and vehicle velocity led to a higher NMPC computational time. However, the soft real-time NMPC implementations did not worsen the RC vehicle performance as the time requirements were fulfilled on average. In conclusion, the successful experiments show the utility of the formulated NMPC controller in low-cost embedded systems and unknown scenarios.

Further work can be done to optimize the programming code developed in the scope of this thesis. Furthermore, to obtain a better performance of the NMPC controller in unknown scenarios, it might be beneficial to create an algorithm that updates the reference trajectory as soon as there is a complete change in the scenario, *e.g.*, when the RC vehicle enters a new room.

References

[1] Sundials. `https://computing.llnl.gov/projects/sundials/`. Accessed: 2020-09-14.

[2] Swig. `http://www.swig.org/`. Accessed: 2020-08-26.

[3] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.

[4] V. Bachtiar, E. C. Kerrigan, W. H. Moase, and C. Manzie. Smoothness properties of the mpc value function with respect to sampling time and prediction horizon. In *2015 10th Asian Control Conference (ASCC)*, pages 1–6, 2015.

[5] R. Bishop. A survey of intelligent vehicle applications worldwide. In *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511)*, pages 25–30, 2000.

[6] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat. Mpc-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3:265–291, 01 2005.

[7] C. Burth. Modelling and identification of a vehicle dynamics model for an rc vehicle, 2018.

[8] M. Diehl and S. Gros. *Numerical Optimization of Dynamic Systems*. 02 2016.

[9] C. Huang, B. Li, and M. Kishida. Model predictive approach to integrated path planning and tracking for autonomous vehicles. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1448–1453. IEEE, 2019.

[10] A. Huber and M. Gerdts. A dynamic programming mpc approach for automatic driving along tracks and its realization with online steering controllers. *IFAC-PapersOnLine*, 50(1):8686–8691, 2017.

[11] J. H. Lee. Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, 9(3):415, 2011.

[12] A. Liniger, A. Domahidi, and M. Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[13] T. Luettel, M. Himmelsbach, and H. Wuensche. Autonomous ground vehicles—concepts and a path to the future. *Proceedings of the IEEE*, 100(Special Centennial Issue):1831–1839, 2012.

[14] MATLAB. *9.7.0.1190202 (R2019b)*. The MathWorks Inc., Natick, Massachusetts, 2019.

[15] M. K. Nasir, R. Md Noor, M. Kalam, and B. Masum. Reduction of fuel consumption and exhaust pollutant using intelligent transport systems. *The Scientific World Journal*, 2014, 2014.

[16] H. B. Pacejka. Modelling of tyre force and moment generation. In B. Jacobson and J. J. Kalker, editors, *Rolling Contact Phenomena*, pages 277–327, Vienna, 2000. Springer Vienna.

[17] J. Park, D. Kim, Y. Yoon, H. Kim, and K. Yi. Obstacle avoidance of autonomous vehicles based on model predictive control. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 223(12):1499–1516, 2009.

[18] R. Quirynen, K. Berntorp, and S. Di Cairano. Embedded optimization algorithms for steering in autonomous vehicles based on nonlinear model predictive control. In *2018 Annual American Control Conference (ACC)*, pages 3251–3256, 2018.

[19] N. Saraf, M. Zanon, and A. Bemporad. A fast nmpc approach based on bounded-variable nonlinear least squares. *IFAC-PapersOnLine*, 51(20):337 – 342, 2018. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[20] G. J. Sutton and R. R. Bitmead. Performance and computational implementation of nonlinear model predictive control on a submarine. In *Nonlinear Model Predictive Control*, pages 461–472. Springer, 2000.

[21] Y. Yoon. Obstacle avoidance for wheeled robots in unknown environments using model predictive control. pages 6792–6797, 07 2008.

[22] Y. Yoon, J. Shin, H. J. Kim, Y. Park, and S. Sastry. Model-predictive active steering and obstacle avoidance for autonomous ground vehicles. *Control Engineering Practice*, 17(7):741 – 750, 2009.