Trajectory Tracking with a Mobile Robot Using Linearization: a Model Predictive Approach

Maxim-Leonid Rezan (6296203), Nicola Visentin (6354815)

***Abstract*—A model predictive controller (MPC) is applied to a simple wheeled robot exploiting linearization along a reference trajectory. The usual MPC formulation is adapted to the linear time-varying (LTV) problem and a stability analysis is performed, leading to two different designs for the controller. An extended Kalman filter (EKF) is developed for controlling the robot when a constant measurements disturbance occurs. Finally, some simulations are performed both in Matlab/Simulink and in Gazebo simulation environment to test the efficiency of the developed MPC.**

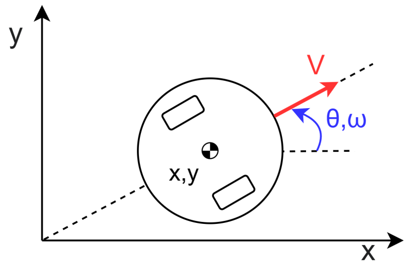

Fig. 1. Unicycle kinematic vehicle model.

I. Introduction

Trajectory tracking for ground vehicles is a critical topic that has gained significant attention, in particular thanks to the recent advancements in autonomous robots and self-driving vehicles [1]. Many control strategies have been developed to face this problem, but among all of them, model predictive control (MPC) proves to be particularly suitable due to its capability of handling constraints, assure good performances and predict the future behavior of the system, which is specially useful in complex dynamic environments or in presence of obstacles [2].
Several models are proposed to describe the dynamics of a ground wheeled vehicle, ranging from very simple kinematic models to more complex representations that also include, for example, tires behavior or aerodynamical interactions.
In this report, we present the design and implementation of a MPC to drive a unicycle robot around a certain reference trajectory. First, the system's model and the control tasks are introduced, secondly we discuss and motivate the MPC design, then a stability analysis is performed and finally we show some simulations results to prove the effectiveness of the regulator.

A. Nonlinear model of the system

A very simple and well-known unicycle kinematic model is used to describe the mobile robot:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{1}$$

where the states (x, y, θ) are respectively the coordinates of the vehicle in a cartesian global reference and the orientation, while the control inputs (v, ω) are the linear velocity and yaw rate, as shown in Figure 1. Despite its simplicity, this model is often used as an approximation of the real behavior of the robot, in particular when inertial effects are low (small accelerations) and physical phenomena such as slipping of the wheels are negligible [3]. Moreover, the nonlinear, non-holonomic nature of the system makes the control task non trivial, requiring more advanced control laws in many scenarios [4]. In our specific case, the choice of this model was mainly driven by the possibility to test the developed MPC controller using Gazebo simulation environment with `turtlebot` robot, which is indeed a unicycle-like vehicle.

B. Linearized model

The control task consists in following a certain known reference trajectory $(x_{\text{ref}}, y_{\text{ref}}, \theta_{\text{ref}})$ generated by a reference control $(u_{\text{ref}}, \omega_{\text{ref}})$. To do so, drawing inspiration from [5], a change of coordinates is performed, linearizing the nonlinear kinematics around this "virtual robot". Defining $\mathbf{x} = [x\ y\ \theta]^T$, $\mathbf{u} = [v\ \omega]^T$ and rewriting (1) in matrix form:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \end{bmatrix} \quad \rightarrow \quad \dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x}, \mathbf{u})$$

Using first order Taylor expansion around the reference trajectory:

$$\dot{\mathbf{x}} - \dot{\mathbf{x}}_{\mathbf{ref}} = \left.\frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}\right|_{\text{ref}} (\mathbf{x} - \mathbf{x}_{\mathbf{ref}}) + \left.\frac{\partial \boldsymbol{f}}{\partial \mathbf{u}}\right|_{\text{ref}} (\mathbf{u} - \mathbf{u}_{\mathbf{ref}})$$

Defining $\mathbf{e} = \mathbf{x} - \mathbf{x}_{\text{ref}}$ and $\mathbf{u_b} = \mathbf{u} - \mathbf{u}_{\text{ref}}$ we can express the dynamics of the error with respect to the reference as:

$$\dot{\mathbf{e}} = A_c\ \mathbf{e} + B_c\ \mathbf{u_b} \tag{2}$$

where

$$A_c = \begin{bmatrix} 0 & 0 & -v_{\text{ref}}\sin(\theta_{\text{ref}}) \\ 0 & 0 & v_{\text{ref}}\cos(\theta_{\text{ref}}) \\ 0 & 0 & 0 \end{bmatrix}$$

$$B_c = \begin{bmatrix} \cos(\theta_{\text{ref}}) & 0 \\ \sin(\theta_{\text{ref}}) & 0 \\ 0 & 1 \end{bmatrix}$$

Finally, in order to implement the MPC, dynamics in (2) is discretized with a sampling time T using a zero-order holder approximation, giving:

$$\mathbf{e}(i+1) = A(i)\mathbf{e}(i) + B(i)\mathbf{u_b}(i) \tag{3}$$

$$A(i) = \begin{bmatrix} 1 & 0 & -Tv_{\text{ref}}(i)\sin(\theta_{\text{ref}}(i)) \\ 0 & 1 & Tv_{ref}(i)\cos(\theta_{\text{ref}}(i)) \\ 0 & 0 & 1 \end{bmatrix}$$

$$B(i) = \begin{bmatrix} T\cos(\theta_{\text{ref}}(i)) & 0 \\ T\sin(\theta_{\text{ref}}(i)) & 0 \\ 0 & T \end{bmatrix}$$

where $i = 0, 1, 2, ..., N_{\text{ref}}$ and N_{ref} is the total number of time instants that we used to define our reference trajectory.
Notice that (3) is a discrete linear time-varying (LTV) system, as v_{ref} and θ_{ref} change at each time instant. It is also important to remark that the linearized system (2) is not suitable for stabilizing the vehicle in a fixed point (x, y, θ) due to lack of controllability, but can be used instead for trajectory tracking problems (i.e. when $(v_{\text{ref}}, \omega_{\text{ref}}) \neq 0$), as in our case [6][4]. The control task reduces to bring the error $\mathbf{e}$ towards zero. Thus, the "real" input $\mathbf{u}$ of the robot will be composed of a feedforward part $\mathbf{u}_{\text{ref}}$ plus a feedback component $\mathbf{u_b}$, which is the one computed by the MPC controller.

II. Model Predictive Control Design

For the general MPC formulation the reader is referred to [7], while the aim of this section is to explain how we adapted and chose the various "ingredients" to design the proposed MPC. As already anticipated, a linear time-varying MPC was developed to control (3) to the origin. In particular, two main scenarios are considered:

- LTV MPC assuming ideal vehicle behavior according to (1).
- LTV MPC with constant disturbance rejection under random input disturbances and measurement noise.

The cost function for the N-steps optimization problem to be solved at each iteration is in the form:

$$J = V_f(\mathbf{e}(N)) + \sum_{k=0}^{N-1} \ell(\mathbf{e}(k), \mathbf{u_b}(k)) \tag{4}$$

with quadratic final cost V_f and quadratic stage cost ℓ:

$$V_f = \frac{1}{2}\mathbf{e}(N)^T P \mathbf{e}(N)$$

$$\ell = \frac{1}{2}\mathbf{e}(k)^T Q \mathbf{e}(k) + \frac{1}{2}\mathbf{u_b}(k)^T R \mathbf{u_b}(k) \tag{5}$$

The choices of the weighting matrices P, Q, R, of the prediction horizon N, as well as all the constraints will be discussed in the following paragraphs.

A. Basic LTV MPC for trajectory tracking

First, an ideal scenario with no disturbances nor noises is assumed. Our LTV MPC formulation is very similar to the "normal" one for linear time-invariant (LTI) systems, with some slight variations due to the time-varying nature of A and B matrices. The N-steps error prediction for the optimization is still expressed as a function of the control history and the initial condition as:

$$\mathbf{X} = T\mathbf{e}(0) + S\mathbf{U}$$

where [1]:

$$\mathbf{X} = [\mathbf{e}(0)\ \mathbf{e}(1)\ ...\ \mathbf{e}(N)\ ]^T \in \mathbb{R}^{3(N+1)}$$

$$\mathbf{U} = [\mathbf{u}_b(0)\ \mathbf{u}_b(1)\ ...\ \mathbf{u}_b(N-1)\ ]^T \in \mathbb{R}^{2N}$$

$$T = \begin{bmatrix} I \\ A(0) \\ A(1)A(0) \\ \vdots \\ A(N-1)A(N-2)\ ...\ A(0) \end{bmatrix}$$

$$S = \begin{bmatrix} 0 & 0 & 0 & ... & 0 \\ B(0) & 0 & 0 & ... & 0 \\ A(1)B(0) & B(1) & 0 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ A(N-1)...A(1)B(0) & & \cdots & & B(N-1) \end{bmatrix}$$

Differently from the LTI case, prediction matrices T and S will change at each iteration. Moreover, our knowledge of the system is limited to N_{ref} time steps, since we only have available $A(i), B(i)$ with $i = 0, ..., N_{\text{ref}}$ matrices, and this means that we will be able to control the robot only for $N_{\text{ref}} - N$ iterations. These two concepts are visually represented in Figure 2.

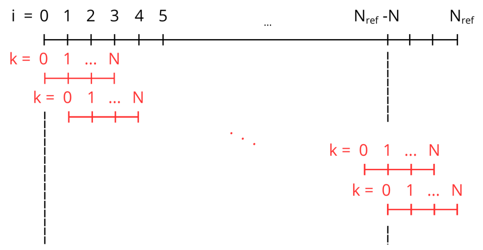

Fig. 2. Black line is the time horizon used to define the reference trajectory. Prediction horizon is shown in red. At each iteration, prediction horizon shifts of one time step, so matrices T and S will change. Also, after $N_{\text{ref}} - N$ iterations, MPC won't be able to predict the system's behaviour anymore.

It's important to notice that all A, B, T and S matrices can be computed and stored offline, since they only depend on the reference, which is known in advance. Thus, the online computational load of the controller is not affected.

[1]Concerning the notation: the MPC in this case works on the error dynamics, so it sees $\mathbf{e}$ as the "state" and $\mathbf{u_b}$ as the "control input". Vectors $\mathbf{X}$ and $\mathbf{U}$ collecting these "states" and "control inputs" time histories are maintained with this notation for consistency with the usual MPC formulation. The same applies to constraint sets $\mathbb{X}$, $\mathbb{U}$, $\mathbb{X}_f$ and to the feasible set $\mathcal{X}_N$.

Similarly, the cost function in (4) can be rewritten in compact form:

$$J = \frac{1}{2}\mathbf{U}^{\mathbf{T}} H \mathbf{U} + (\mathbf{h}\ \mathbf{e}(0))^T \mathbf{U} + const$$

As for the prediction matrices, also the cost matrices H and $\mathbf{h}$ (whose definition is here omitted) will change at each iteration, since the cost function depends on $\mathbf{X}$, but $\mathbf{X}$ is expressed as a function of $\mathbf{U}$ through T and S. However, everything can be computed and stored offline.

The constraints for the optimization problem, as presented in chapter 2.2 of [7], are:

$$\mathbf{e}(k) \in \mathbb{X}\ for\ k = 0, .., N$$

$$\mathbf{u_b}(k) \in \mathbb{U}\ for\ k = 0, .., N-1$$

Concerning the input constraints, we know that the velocities of our robot are limited:

$$v \in [v_{min},\ v_{max}]$$

$$\omega \in [\omega_{min},\ \omega_{max}]$$

Since these constant limits apply to the "real" input $\mathbf{u}$, we need to translate them into bounds on $\mathbf{u_b}$. This is quite trivial, as we just need to impose, for $k = 0, ..., N-1$:

$$\begin{bmatrix} v_{min} \\ \omega_{min} \end{bmatrix} - \mathbf{u}_{\text{ref}}(k) \le \mathbf{u_b}(k) \le \begin{bmatrix} v_{max} \\ \omega_{max} \end{bmatrix} - \mathbf{u}_{\text{ref}}(k)$$

which can be written in matrix form as:

$$A_{in,1}\mathbf{U} \le \mathbf{b_{in,1}}$$

Additionally, the error was bounded to $\mathbf{e_{min}} \le \mathbf{e}(k) \le \mathbf{e_{max}}$ to make sure that the system stays close to the reference and the linearization remains a good approximation. Again, this constraint can be written in matrix form:

$$A_{in,2}\mathbf{U} \le \mathbf{b_{in,2}}$$

Vector $\mathbf{b_{in,1}}$ will change during the iterations (apart from the physical constraints, it also depends on the reference). Matrix $A_{in,2}$ and vector $\mathbf{b_{in,2}}$ will change as well (even if the bounds are constant in this case, $\mathbf{e}(k)$ is related to $\mathbf{u_b}(k)$ through $A(k)$ and $B(k)$). In any case, everything can be preallocated offline.

Cost matrices Q and R were chosen as:

$$Q = \begin{bmatrix} 10/e^2_{1,max} & 0 & 0 \\ 0 & 10/e^2_{2,max} & 0 \\ 0 & 0 & 10/e^2_{3,max} \end{bmatrix}$$

$$R = \begin{bmatrix} 1/v^2_{max} & 0 \\ 0 & 1/\omega^2_{max} \end{bmatrix}$$

These values were found experimentally tying to achieve good tracking performances with a relatively limited control effort. The normalization is there to account for the different order of magnitude of the various quantities.
Prediction horizon length $N = 20$ was tuned empirically as well, with the aim of having a feasible optimization problem but with the smallest possible computational load. Notice that since we are just trying to keep the robot on a reference path, the prediction horizon can be quite small, since **ideally** the actual vehicle would perfectly follow the behavior of the reference "virtual" one. Also, as shown in Figure 2, having a shorter receding horizon allows for controlling the system over a larger part of the reference trajectory.

Finally, two different choices were made for the terminal set $\mathbb{X}_f$ and the terminal weight matrix P, with the goal of assuring stability of the closed-loop. They will be discussed in section III.

The effects of varying these parameters will be discussed in section IV.

B. LTV MPC with disturbance rejection

Now the assumption of an ideal environment is relaxed. Due to the simplicity of the model, we still assume that the full state measurement is available, however, this measurement is affected by noise and include a constant disturbance offset in the heading θ. Additionally, random disturbances are introduced into the control input signals. Thus, system (1) becomes:

$$\begin{cases} \dot{x} = (v + w_1)\cos(\theta) \\ \dot{y} = (v + w_1)\sin(\theta) \\ \dot{\theta} = \omega + w_2 \end{cases} \quad \rightarrow \quad \dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x}, \mathbf{u}, \mathbf{w})$$

$$\begin{cases} y_1 = x + \nu_1 \\ y_2 = y + \nu_2 \\ y_3 = \theta + d + \nu_3 \end{cases} \quad \rightarrow \quad \mathbf{y} = \boldsymbol{h}(\mathbf{x}, d, \boldsymbol{\nu})$$

where $\mathbf{w} = [w_1\ w_2]^T \in \mathbb{R}^2$ are the input disturbances, $\mathbf{y} \in \mathbb{R}^3$ are the noisy measurements, $d \in \mathbb{R}$ is the constant unknown offset on the heading measurement and $\boldsymbol{\nu} = [\nu_1\ \nu_2\ \nu_3]^T \in \mathbb{R}^3$ is the output noise. In particular we assume:

- $\mathbf{w} \sim \mathcal{N}(0, Q_w)$ is a zero-mean, gaussian, white random disturbance with covariance $\mathbb{E}[\mathbf{w}\mathbf{w}^T] = Q_w \in \mathbb{R}^{2x2}$
- $\boldsymbol{\nu} \sim \mathcal{N}(0, R_{ob})$ is a zero-mean, gaussian, white random noise with covariance $\mathbb{E}[\boldsymbol{\nu}\boldsymbol{\nu}^T] = R_{ob} \in \mathbb{R}^{3x3}$

The general architecture and design choices of the controller are the same as in the previous paragraph, however now the constant offset d obviously causes the MPC to drive the robot towards a wrong trajectory. Drawing inspiration from the lectures, the idea is to extend the system and build a filter to estimate d. The augmented system, also including the offset dynamics $\dot{d} = 0$, is:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}) \\ 0 + w_p \end{bmatrix} \quad \rightarrow \quad \dot{\mathbf{x}}_{\mathbf{a}} = \tilde{\boldsymbol{f}}(\mathbf{x_a}, \mathbf{u}, \mathbf{w_a})$$

$$\mathbf{y} = \begin{bmatrix} x \\ y \\ \theta + d \end{bmatrix} + \boldsymbol{\nu} \quad \rightarrow \quad \mathbf{y} = \tilde{\boldsymbol{h}}(\mathbf{x_a}, \boldsymbol{\nu})$$

where $\mathbf{x_a} = [x\ y\ \theta\ d]^T \in \mathbb{R}^4$ is the augmented state, $\mathbf{w_a} = [w_1\ w_2\ w_p]^T \in \mathbb{R}^3$ is the augmented disturbance vector. Notice that an artificial disturbance w_p with covariance $Q_p \in \mathbb{R}$ is added to the offset dynamics: this will allow the filter to converge towards an estimation of d. The new extended disturbance covariance is thus:

$$\mathbb{E}(\mathbf{w_a}\mathbf{w_a}^T) = \begin{bmatrix} Q_w & 0 \\ 0 & Q_p \end{bmatrix} = Q_{ob} \in \mathbb{R}^{3\text{x}3}$$

Since the system is nonlinear and a unique linearization near an equilibrium point is not available, a nonlinear extended Kalman filter (EKF) is used to find the estimation $\hat{\mathbf{x}}_\mathbf{a}$ of the states and the offset disturbance. Obviously, the observer also allows to filter out $\boldsymbol{\nu}$ and $\mathbf{w}$. The estimated state $\hat{\mathbf{x}}$ is then used in the MPC in place of the noisy and offset measurement $\mathbf{y}$. The EKF can also provide an estimation $\hat{d}$ of the offset disturbance. The scheme is shown in Figure 3.

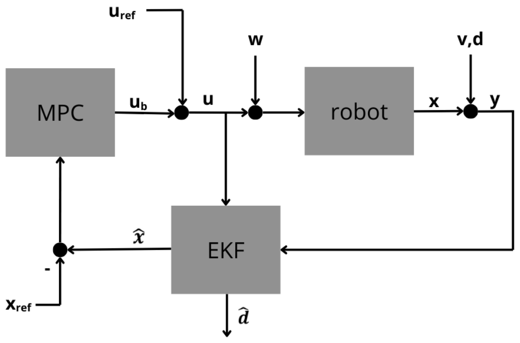

Fig. 3. Control scheme with disturbance rejection.

The EKF derivation is not explained in detail. However, the filter is:

$$\dot{\hat{\mathbf{x}}}_\mathbf{a} = \tilde{\boldsymbol{f}}(\hat{\mathbf{x}}_\mathbf{a}, \mathbf{u}) + K_{ob}(\mathbf{y} - \tilde{\boldsymbol{h}}(\hat{\mathbf{x}}_\mathbf{a}))$$

with Kalman gain matrix:

$$K_{ob} = P_{ob} C_a^T R_{ob}^{-1}$$

and P_{ob} solution of the Riccati equation:

$$\dot{P}_{ob} =$$
$$= A_a P_{ob} + P_{ob} A_a^T - P_{ob} C_a^T (M_a R_{ob} M_a^T)^{-1} C_a P_{ob} + L_a Q_{ob} L_a^T$$

where:

$$A_a = \left.\frac{\partial \tilde{\boldsymbol{f}}}{\partial \mathbf{x_a}}\right|_{\hat{\mathbf{x}}_\mathbf{a}} \quad C_a = \left.\frac{\partial \tilde{\boldsymbol{h}}}{\partial \mathbf{x_a}}\right|_{\hat{\mathbf{x}}_\mathbf{a}} \quad L_a = \left.\frac{\partial \tilde{\boldsymbol{f}}}{\partial \mathbf{w_a}}\right|_{\hat{\mathbf{x}}_\mathbf{a}} \quad M_a = \left.\frac{\partial \tilde{\boldsymbol{h}}}{\partial \boldsymbol{\nu}}\right|_{\hat{\mathbf{x}}_\mathbf{a}}$$

The covariances were simply chosen as $Q_w = diag(1,1)$, $R = diag(1,1,1)$ to impose a balanced reliability of the system model and of the measurements and $Q_p = 10$ was selected to allow a fast but accurate estimation of d.

III. Stability analysis

In this section we'll discuss the analysis developed for choosing the terminal set $\mathbb{X}_f$ and the final cost V_f for the MPC optimization problem in order to assure asymptotic stability of the closed-loop with the developed controller. In particular, two strategies were implemented, tested and compared: a very trivial choice of a null final weight with terminal set being the origin and another case where $\mathbb{X}_f$ is taken as a maximal invariant set for all the configurations.

A. $\mathbb{X}_f = \{\mathbf{0}\}$ *and null final cost*

As suggested in section 2.5.6 of [7], for a linear time-varying system as (3), with cost function like (4), the terminal set can be chosen as $\mathbb{X}_f = \{\mathbf{0}\}$ and the final cost $V_f = 0$ for all the iterations. Making so, the Lyapunov decrease in Assumption 2.33 (a) is trivially satisfied, while 2.33 (b) is met as well by choosing for example $\alpha_1 = \ell$ (which is indeed a $\mathcal{K}_\infty$ function) and any α_f (since $V_f = 0 \quad \forall i$). Moreover, $A(i)$ and $B(i)$ matrices in our problem can be bounded from above, as well as Q and R (that are constant) and the system is stabilizable in the reference trajectory, as explained in subsection I-B and [4]: Assumption 2.37 is satisfied. Finally, (3), $\ell(\mathbf{e}, \mathbf{u_b}, i)$, V_f are continuous and null for $(\mathbf{e}, \mathbf{u_b}) = (\mathbf{0}, \mathbf{0})$ (Assumption 2.25) and all constraints sets are close, compact and containing the origin (Assumption 2.26). All assumptions for Theorem 2.39 are satisfied, so the asymptotic stability of our time-varying MPC is proven.

Obviously, this choice of $\mathbb{X}_f$ is very restrictive, and makes so that the MPC optimization problem can be solved for a very limited set of initial conditions $\mathbf{e}(0)$ that are close to the origin or for a large prediction horizon. In general, we expect the admissible set $\mathcal{X}_N$ to be very small. However, if the initial configuration of the robot is close enough to the initial point of the reference trajectory, and if the behavior of the vehicle is well represented by the linearized kinematic model (small curvature of the reference path, small discretization step, low speeds, no relevant inertial effects, etc...) this approach can still be suitable.

B. $\mathbb{X}_f$ *as maximal positive invariant set for all* i

Drawing inspiration from [8], another approach was used to ensure asymptotic stability of the controller. When a MPC is applied to a LTI system, a typical choice for $\mathbb{X}_f$ is the maximal invariant set within which the MPC "behaves" as an LQR (i.e. unconstrained, infinite time optimal control problem). Since our model is LTV, the idea is to choose $\mathbb{X}_f$ as the maximal invariant set that is invariant for all the linearizations around the reference trajectory. To do so, a terminal set is computed for each linearization and all these sets are combined together according to [8], using MPT3 toolbox [9].
On the other hand, when dealing with a LTI system, the terminal cost is usually taken as $V_f = \frac{1}{2}\mathbf{x}(N)^T P \mathbf{x}(N)$, where P is the Riccati matrix solution of the unconstrained, infinite time optimal control problem associated with (5), that has optimal cost $J_\infty^{unc} = \frac{1}{2}\mathbf{x}^T P \mathbf{x}$. For our LTV system, the idea is to determine P by finding an upper bound of the infinite horizon cost for all the i-th linearizations:

$$\frac{1}{2}\mathbf{e}^T P \mathbf{e} \geq \max_{\text{i}} \left\{ \frac{1}{2}\mathbf{e}^T P(i) \mathbf{e} \right\} \quad subj. \ to \ \ \mathbf{e} \in \mathbb{X}_f$$

where $P(i)$ is the Riccati matrix associated to the infinite time LQR problem for the i-th linearization, while $\mathbf{e} \in \mathbb{X}_f$ is there because this cost is evaluated in an unconstrained situation (i.e. when we are inside the terminal set). Actually, since $\mathbb{X}_f$ is convex, we only need to consider its vertices:

$$\frac{1}{2}\mathbf{e}^T P \mathbf{e} \geq \max_{\text{i}} \left\{ \frac{1}{2}\mathbf{e}^T P(i) \mathbf{e} \right\} \quad subj. \ to \ \ \mathbf{e} \in \mathcal{V}$$

where $\mathcal{V} = \{\mathbf{v}_1,\ \mathbf{v}_2,\ ...\}$ defines the vertices of $\mathbb{X}_f$. This can be further simplified by introducing a change of variables, as suggested in [8], and the final problem to be solved is:

$$\frac{1}{2}\mathbf{e}^T P\mathbf{e} \geq \max_{\text{i}} \{\mathbf{w}^T\mathbf{w}\} \quad subj.\ to\ \ \mathbf{w} \in \mathcal{W}$$

where $\mathcal{W} = \{L\mathbf{v}_1,\ L\mathbf{v}_2,\ ...\}$ and $L(i)$ is the lower triangular Cholesky decomposition of $P(i)$.

With this choice of $\mathbb{X}_f$ and P, the proposed MPC is proven in [8] to be asymptotically stable, if:

- Q, R, V_f are symmetric positive definite
- $\mathbb{U}$, $\mathbb{X}$, $\mathbb{X}_f$ are closed and contain the origin
- $\mathbb{X}_f \subset \mathbb{X}$ is control invariant

which are all satisfied in our case.

C. Numerical tests and comparison

In this paragraph some numerical results are discussed to analyze and compare the two choices of the terminal set. Unless otherwise specified, trajectory n1 proposed in subsection IV-A was used. It is also worth mentioning that in the procedure to compute $\mathbb{X}_f$ in the LQR terminal set scenario we didn't actually iterate through all the possible linearizations (due to memory issues), but this is somehow fine, thanks to the nature of the chosen reference (v_{ref} and θ_{ref} with small variations and periodic).

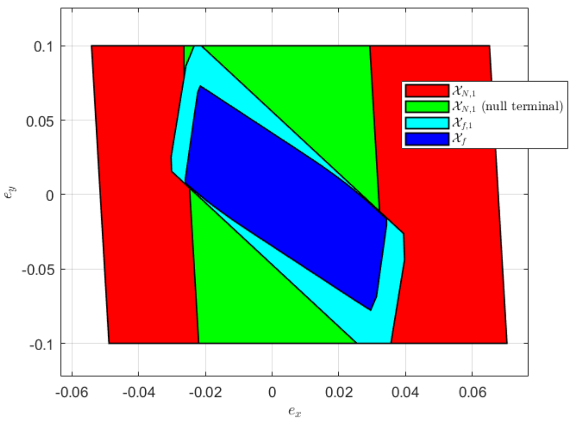

Fig. 4. Green: $\mathcal{X}_6$ for the first iteration associated to the first design choice. Red: $\mathcal{X}_6$ for the first iteration associated to the second design choice. Cyan: $\mathbb{X}_f$ for the first iteration associated to the second design choice. Blue: final $\mathbb{X}_f$ for the second design choice.

First, both MPCs were run with the same parameters (except for $\mathbb{X}_f$ and V_f, obviously) and a receding horizon $N = 6$. The admissible sets $\mathcal{X}_N$ associated with the first iteration[2] are plotted in Figure 4 for comparison (green for the origin terminal set case, red for the LQR terminal set case). As expected, when $\mathbb{X}_f = \{\mathbf{0}\}$ a smaller admissible set arises, meaning that the optimization problem will be harder to solve (we need higher N or error close to the origin). This is further confirmed by reducing the prediction horizon to $N = 5$: now the MPC optimization with $\mathbb{X}_f = \{\mathbf{0}\}$ is infeasible with the selected data, while the other MPC is still observed to work, stabilizing the system. This shows, as already anticipated, that the first design choice is simpler but more restrictive.

[2]Actually their 2D projections on (e_x, e_y) plane.

Let's now focus on the second design choice, i.e. $\mathbb{X}_f$ as the maximal LQR invariant set. Figure 4 is also showing the final terminal set $\mathbb{X}_f$ (blue) and a specific terminal set associated to the first linearization (cyan). Obviously, $\mathbb{X}_f \subseteq \mathbb{X}_{f,1}$ as $\mathbb{X}_f$ is the maximal invariant set which is invariant for *all* the linearizations. For the same reason, we can imagine that the more complex the trajectory is (i.e. v_{ref} and θ_{ref} vary a lot), the smaller $\mathbb{X}_f$ will be. This is shown in Figure 5, where the reference speed and yaw angle vary more and more from trajectory n1 to trajectory n3 (see their definition in subsection IV-A).

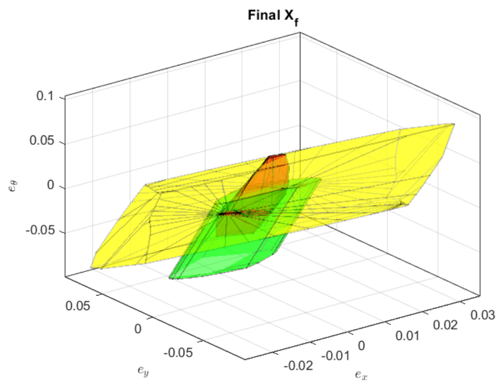

Fig. 5. $\mathbb{X}_f$ sets for three different reference trajectories. Yellow: trajectory n1. Green: trajectory n2. Red: trajectory n3.

Finally, we can imagine that X_f also depends on Q and R matrices, as they influence the LQR solution. Figure 6 and Figure 7 show the resulting terminal sets for trajectory n1 with different values of the weighting matrices.

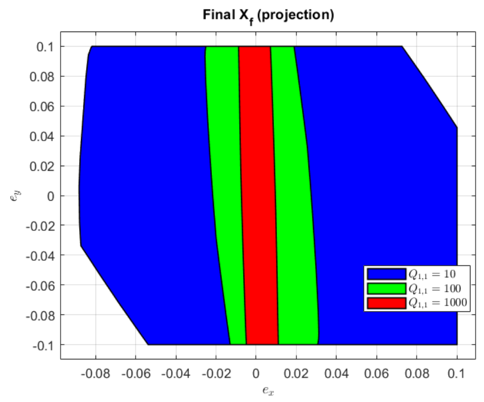

Fig. 6. Projections of $\mathbb{X}_f$ sets for different values of Q.

As we can see, if the penalization on the error increases, the controller will tend to "bring" $\mathbf{e}(k)$ to zero as soon as possible along the prediction horizon N. This results in a more aggressive control and translates in a smaller terminal

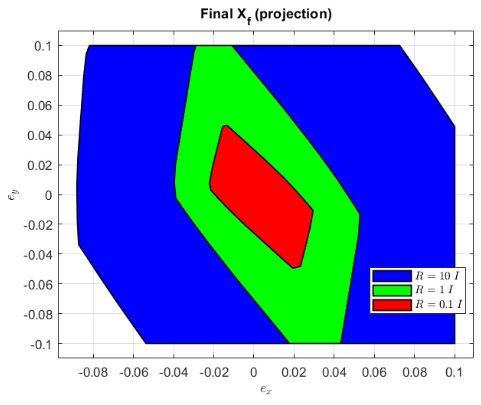

Fig. 7. Projections of $\mathbb{X}_f$ sets for different values of R.

constraint region. In particular, in Figure 6 the weight $Q_{1,1}$ on e_x was increased, and this results in a terminal set which is stricter and stricter in the e_x direction. Similar considerations hold for input penalization matrix R: this time, the larger it is, the more the inputs are penalized, meaning that the MPC will be less aggressive. This translates in a larger set $\mathbb{X}_f$.

IV. Simulations and Results

In this section, some numerical simulation results are presented to evaluate the performances of the developed MPC. First, we briefly explain how the reference trajectories are generated. Then we show some results for both the disturbance-free scenario and for the MPC with disturbance rejection. Finally, some simulations in the Gazebo environment using ROS are presented as well.

A. Reference trajectory

In general, the linearized MPC proposed in this report relies on an approximation of the nonlinear system. We can intuitively imagine that this approach is not effective if the discretization is poor or if the trajectories are very complex, i.e. v_{ref} changes rapidly or the curvature is severe (see (3)).

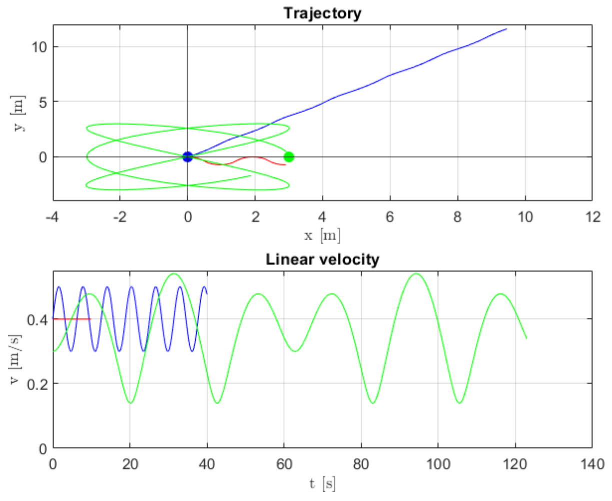

Fig. 8. Reference trajectories and corresponding reference linear velocities. Blue: trajectory n1. Red: trajectory n2. Green: trajectory n3.

With this idea in mind, we decided to test the controller on a suitable trajectory (trajectory n1). However, some simulations were also performed with different paths for comparison (trajectories n2 and n3). All references are shown in Figure 8. To generate $\mathbf{x}_{\text{ref}}$ and $\mathbf{u}_{\text{ref}}$ two approaches were used:

- Traj. n1, n2: simulate a certain imposed control $\mathbf{u}_{\text{ref}}(t)$ with ode45 and save the corresponding $\mathbf{x}_{\text{ref}}(t)$.
- Traj. n3: impose a certain geometrical path $(x, y)_{\text{ref}}$ and derive $\theta_{\text{ref}}(t)$, $\mathbf{u}_{\text{ref}}(t)$ analytically from (1) as:

$$\theta = \arctan \frac{\dot{y}}{\dot{x}} \qquad v = \sqrt{\dot{x}^2 + \dot{y}^2} \qquad \omega = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2}$$

The timestep used for the discretization was $T = 1/30\ s$, in prevision of implementing the MPC in ROS with a frequency of 30 Hz.

B. LTV MPC without disturbances

First, the "basic" MPC formulation presented subsection II-A is tested in Matlab. Weighting matrices Q, R and the prediction horizon N are the ones mentioned in subsection II-A, while the terminal set $\mathbb{X}_f$ and final weight matrix P are chosen according to the second design choice in subsection III-B, which proved to be less restrictive. The constraints for the optimization problem are $|e_j| \leq 0.5\ m\ for\ j = 1, 2, 3$ to assure a small error with respect to the reference, while v_{max}, v_{min} and $\omega_{max}, \omega_{min}$ are chosen according to the trajectory (for trajectory n1: $v \in [0.2,\ 0.55]\ m/s$ and $\omega \in [-0.2,\ 0.2]\ rad/s$).
Two tests are performed, with two different initial configurations of the robot. Figure 9 shows the resulting trajectories: as we can see, provided that the optimization is feasible at each time step (in fact, for test 2, the prediction horizon N was increased[3], because the initial error was higher), the choice of $\mathbb{X}_f$ and V_f assures the stability of the controller (Figure 10 shows the evolution of the error $\mathbf{e}$).

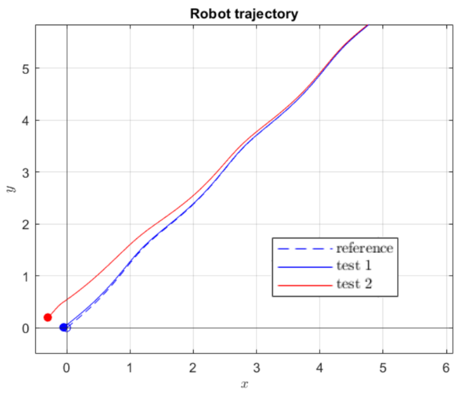

Fig. 9. Tracking problem with different initial conditions: trajectory.

As we can notice from Figure 11[4], that shows the robot control inputs $\mathbf{u}$, if the initial condition $\mathbf{e}(0)$ is far from the origin (and this happens for the first few iterations), the resulting

[3]This also increases the computational load.
[4]Some of the plots has been focused on a shorter time window just to have a clearer visualization.

control is very aggressive, reaching saturation. To overcome this, we tried to increase the penalization on $\mathbf{u}_b$ and reduce the penalization on the error, increasing R by a factor of 100 and reducing Q by 10 times.

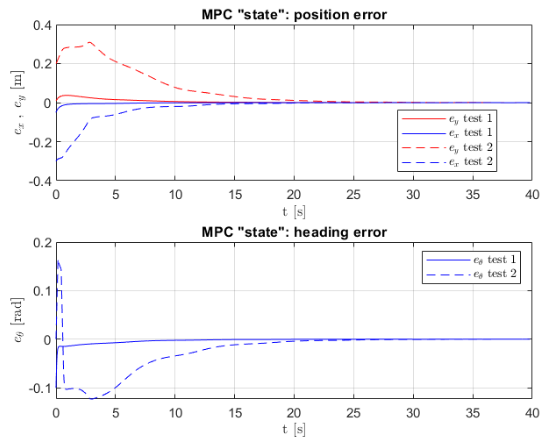

Fig. 10. Tracking problem with different initial conditions: error.

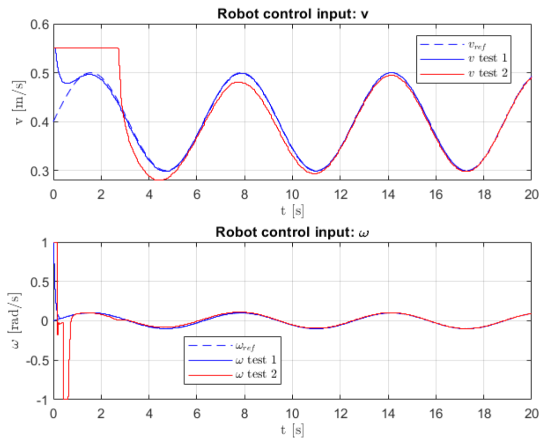

Fig. 11. Tracking problem with different initial conditions: control inputs.

Now, as shown in Figure 12, the control action is less severe as expected, but the robot takes more time to converge to the reference path (Figure 13).

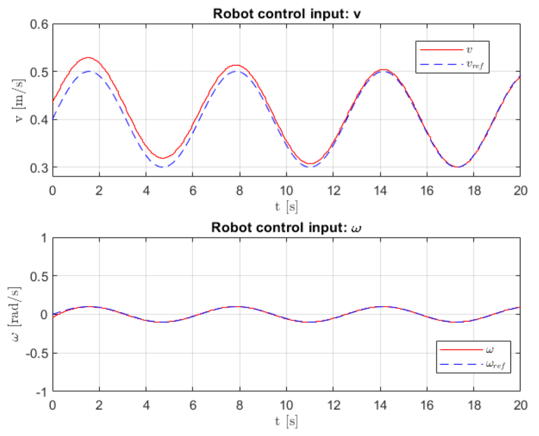

Fig. 12. Effect of increasing R and reducing Q: control inputs.

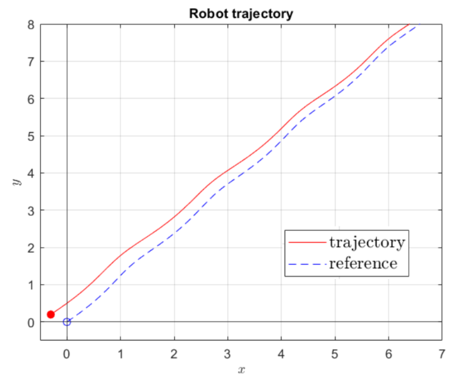

Fig. 13. Effect of increasing R and reducing Q: trajectory.

Finally, the last test was repeated, but without imposing the stability condition (i.e. $\mathbb{X}_f$ was removed). The controller, in this case, is not able to stabilize the system, as shown in Figure 14, where the states x and y of the vehicle are plotted.

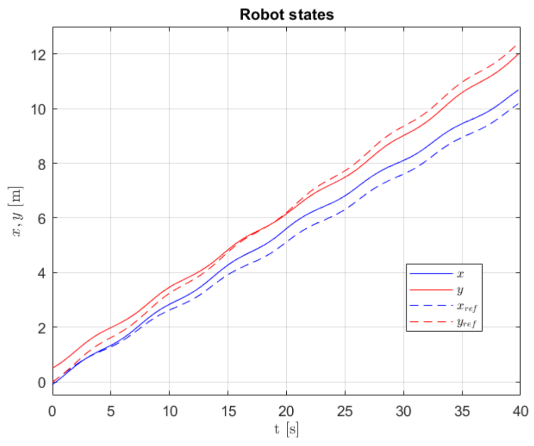

Fig. 14. Simulation without the stability condition.

C. LTV MPC with disturbance rejection

The scheme proposed in Figure 3 was implemented in Simulink. Random noises and input disturbances were introduced as mentioned in subsection II-B, as well as a constant offset disturbance $d = 0.1 \ rad$. The rest of the MPC parameters were left as in test 1 presented in subsection IV-B, apart for matrices Q and R that was changed to make the controll a bit less aggressive. The EKF was initialized with an error on the real initial position of the robot of $0.5 \ m$ for $\hat{x}$ and $\hat{y}$ and of $0.2 \ rad$ for $\hat{\theta}$. The initial guess of the filter for the offset d was instead $\hat{d} = -0.1 \ rad$.

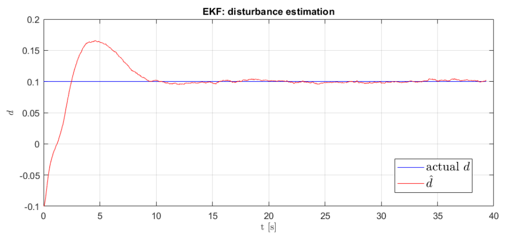

Fig. 15. EKF: disturbance estimation.

Results of the simulations show that the controller still manages to stabilize all states on the reference trajectory (Figure 16), while Figure 15 depicts the filter capability of correctly estimate the constant offset d. Figure 16 also shows the performances of the observer: apart from filtering out the noise, it's worth noticing from the third plot how the estimation of θ (red line) passes from being close to the wrong measurement affected by the offset (green line) to the actual real state (blue line), as the estimation of d improves.

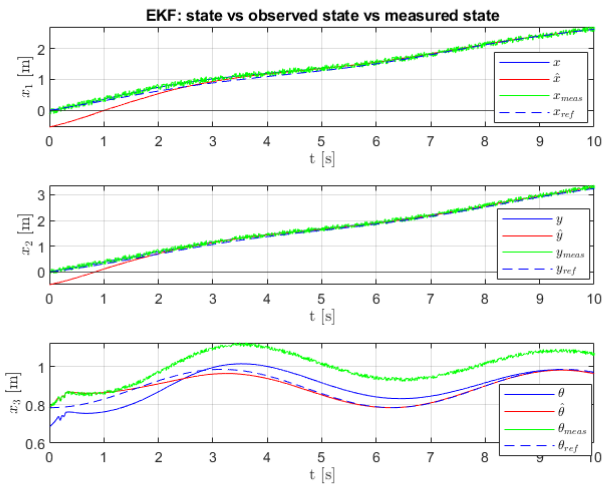

Fig. 16. MPC with disturbance rejection: state vs estimated state vs measured state vs reference state.

D. LTV MPC in a realistic simulation environment

Up to now, the MPC controller was only simulated on a nonlinear kinematic system given by the unicycle model. This doesn't account for many factors that are present in the physical world, like the inertia of the robot or the possible slip of the wheels. As we already said, if the velocities are small, the curvature is not too severe and the accelerations are not so big, then the proposed controller should still be able to achieve its task. In order to test this, the Simulink model used in subsection IV-C was adapted to communicate with Gazebo simulation environment through ROS, and a `turtlebot` unicycle-like robot was controlled with the proposed MPC.

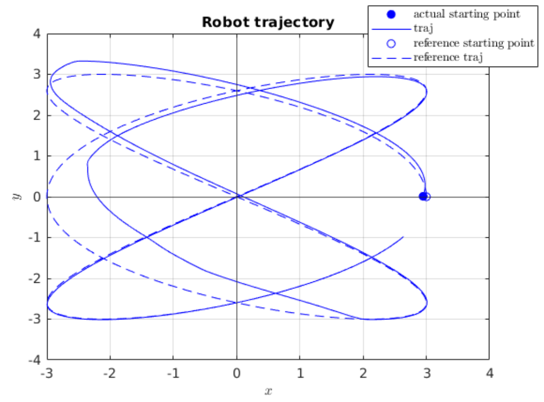

Fig. 17. Simulation in ROS and Gazebo: trajectory.

Trajectory n3 was chosen as the reference[5], so that it was possible to test a more complex and interesting path. Finally, different parameters of the MPC were adjusted to adapt to the new scenario. The outcome of the simulation is shown in Figure 17 and Figure 18. Obviously the tracking accuracy is worse than the previous tests in Matlab, both due to the more complex reference path and to the different model of the robot. In particular, two "critical points" can be recognized: at the very beginning (most likely due to the inertial effect at the start[6] that the MPC is not able to predict), and after more of less 60 seconds (a loss of data was observed here, mainly due to delays in the communication between ROS and Simulink).

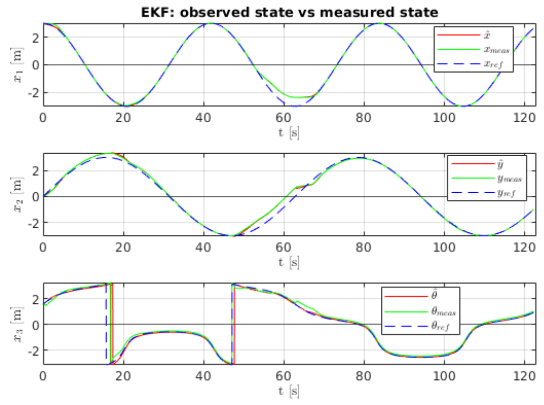

Fig. 18. Simulation in ROS and Gazebo: states.

V. Conclusions

A MPC controller was developed to address trajectory tracking of a reference path in a simple, low speed scenario. A unicycle kinematic system was chosen as model for the MPC, while a linearization along the reference was performed to maintain a linear structure of the problem. The usual architecture and stability conditions of the LTI MPC were adapted to suit the time-varying case and simulation were performed to test the effectiveness of the design.

In general, the controller was observed to work, even in presence of disturbances or in a real-world scenario. Obviously, as the trajectory gets more complex or the discrepancy between the simple kinematic model and the real robot gets bigger, the limits of this approach arise. In that case, more refined control strategies, such as nonlinear MPC [10] should be adopted, at cost of an higher online computational load. One of the advantages of the proposed design, in fact, is the possibility to maintain a contained computational effort.

Topics as model predictive control and trajectory tracking are really vast and popular and find a lots of applications, so for sure many further improvements could be done to the work presented here, for example a comparison with the performances of other controllers (NMPC, LQR, etc...), the integration of a more refined vehicle model (e.g. including dynamics) or a better tuning of the parameters in the proposed MPC through a big number of simulations over many different scenarios.

[5]Also trajectory n1 was tested, giving a better tracking performance and very similar results to the ones obtained in Matlab.

[6]The `turtlebot` had $v = 0$ at the beginning.

References

[1] L. Li, J. Li, and S. Zhang, "Review article: State-of-the-art trajectory tracking of autonomous vehicles," *Mechanical Sciences*, vol. 12, no. 1, pp. 419–432, 2021.

[2] S. Yu, M. Hirche, Y. Huang, H. Chen, and F. Allgöwer, "Model predictive control for autonomous ground vehicles: a review," *Autonomous Intelligent Systems*, vol. 1, 2021.

[3] G. Raffo, G. K. Gomes, J. E. Normey-Rico, C. R. Kelber, and L. B. Becker, "A predictive controller for autonomous vehicle path tracking," in *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 1, 2009, pp. 92–102.

[4] d'Andréa-Novel, Brigitte and Thorel, Sylvain, "Control of non holonomic or under-actuated mechanical systems: The examples of the unicycle robot and the slider," *ESAIM: COCV*, vol. 22, no. 4, pp. 983–1016, 2016.

[5] X. Song, Y. Shao, and Z. Qu, "A vehicle trajectory tracking method with a time-varying model based on the model predictive control," *IEEE Access*, vol. 8, pp. 16 573–16 583, 2020.

[6] C. Samson and K. Ait-Abderrahim, "Feedback control of a nonholonomic wheeled cart in cartesian space," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 1136–1141 vol.2.

[7] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2008.

[8] P. F. Lima, J. Mårtensson, and B. Wahlberg, "Stability conditions for linear time-varying model predictive control in autonomous driving," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 2775–2782.

[9] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *Proc. of the European Control Conference*, Zürich, Switzerland, July 17–19 2013, pp. 502–510.

[10] M. A. Henson, "Nonlinear model predictive control: current status and future directions," *Computers & Chemical Engineering*, vol. 23, no. 2, pp. 187–202, 1998.