2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Safe and Online MPC for Managing Safety and Comfort of Autonomous Vehicles in Urban Environment

Charles Philippe[1,2], Lounis Adouane[1], Benoît Thuilot[1], Antonios Tsourdos[2] and Hyo-Sang Shin[2]

Abstract— **In this paper is presented a linear MPC controller design for autonomous cars navigation. It combines both the lateral and longitudinal control. The MPC cost function has been designed to account for human driving behaviours, i.e., it smoothes out coarse reference trajectories. Furthermore, a safety monitoring module has been implemented. It computes an estimated time before reaching an unacceptable situation (w.r.t. comfort constraints and tracking performance) under the current tracking conditions. The overall benefit of this controller is to guarantee trajectory smoothness while outputting information on its performance. This information will later be used to re-plan safe trajectories in dynamic environments. The proposed linear MPC controller has been tested in a typical urban scenario based on a realistic simulator.**

I. INTRODUCTION

The field of mobile robotics for passenger transportation has been very active in the recent years. Many advances have been done and highlighted by challenges such as the *DARPA Grand Challenge* and more recently by the *Grand Cooperative Challenge*, as well as advances of companies such as Volvo, Uber, Google or Tesla [1]. Recent events have highlighted the necessity to ensure very high levels of reliability in the algorithmic developments to deal with uncertainties in the environment. There is also a necessity of foreseeing and preventing problems that could arise from sensors and actuators faults or inaccuracies. This paper brings a contribution to solving these problems for trajectory tracking.

Model Predictive Control (MPC) is now a widely used and performant technique for optimal trajectory tracking [2], [3] and/or optimal trajectory generation [4]. It allows to cater for future events and the predictive nature often helps generating smoother control signals when the MPC is used as a controller.

For all the above mentioned applications, a linear MPC formulation has to be used, mainly for computational complexity reasons. This formulation introduces limitations in the way the problem can be formulated, since it ultimately has to come down to a Quadratic Programming (QP) formulation. As a consequence, the cost functions used in the nonlinear formulations cannot always be transformed into a corresponding linear formulation for real time implementation.

Other interesting works have moved towards human-like behaviour, such as in [5] for a plannning algorithm, showing good results. Such an approach potentially allows to limit the difference between humans and machines in terms of perceived behaviour and could ultimately allow the passengers to feel less uncomfortable. The approach shown in this paper is to include elements of "smoothness" in the trajectory tracking algorithm (mimicking human behaviour) while ensuring safety and precision. Guaranteeing smoothness of the car trajectory at the controller level additionally means it is not needed to tackle this problem when designing the trajectory planner. Thus, some complexity is removed from the planner and other elements can be taken into account, such as risk minimization.

Some recent planning algorithms have introduced such functions to assess the performance of the subsequent trajectory controller in order to compute a maximal safe speed (see [6] for an offroad application). It has the benefits of ensuring a given tracking performance. The main risks of failure to guarantee tracking performances are actuator saturation and tracking precision (due mainly to disturbances and model inacurracies).

This paper follows up a first contribution for an architecture design including guidance and control presented in [7]. It goes further by linearizing the MPC algorithm, proposing a safety monitoring module and testing it in a realistic simulation environment using the ROS (Robot Operating System) middleware.

The approach proposed in this paper focuses on improving the attainable tradeoff between smoothness of the control signal and tracking performance. This objective has been reached using a linear MPC formulation that gives a smoother control signal than similar works, and a risk monitoring module.

The remainder of this paper is organized as follows. Section II explains the reasons behind the architecure and controller choice, and describes the design of the controller. The characteristics and performance of the proposed controller are assessed in section III through experiments on a realistic simulation software. A conclusion and prospects are given in section IV.

II. PROPOSED ARCHITECTURE

A. Architecture design

The proposed MPC algorithm is intended to be part of a common Guidance, Navigation and Control (GNC) architecture. The navigation layer is not detailed here since it is not the focus of the presented works. The aim is to propse a versatile architecture for navigation of urban vehicles. This architecture should ease the support of convoy navigation and ensure safety. The proposed MPC algorithm is designed as

[1] Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, F-63000 Clermont-Ferrand, France name.surname@uca.fr
[2] Cranfield University, Cranfield, United Kingdom name.surname@cranfield.ac.uk

a trajectory controller for both the lateral and longitudinal axis. The lateral control signal generated by the MPC feeds a low-level yaw-rate controller that controls the car steering. Such a low level controller has been shown in [7]. The MPC controller is coupled with a safety monitoring algorithm that computes a risk associated to the current tracking situation: reference trajectory, desired speed and predicted performance of the MPC controller. This risk is computed based on the following criteria:

- degree of MPC model accuracy
- predicted lateral error

These two criteria give an indication of the tracking performance that can be expected. they can also be used to gobtain an estimation of the likelihood of failing to track accurately the reference trajectory. The general architecture is shown in Fig. 1.

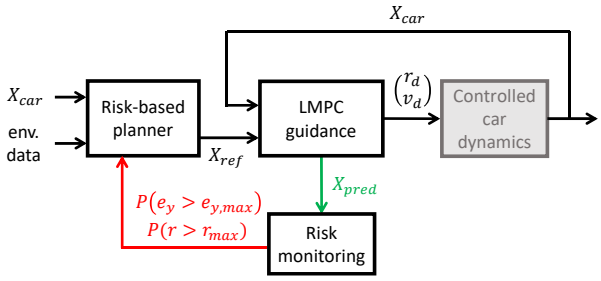

Fig. 1. Planning and risk monitoring architecture

B. MPC controller

A linear MPC controller has been designed. In an earlier publication [7], a nonlinear MPC controller has been presented and tested in simulations against a widely used trajectory controller [8] and a versatile Lyapunov function based nonlinear control law [9] that supports trajectory tracking, mobile target tracking and waypoint-based navigation. This controller showed a significant increase in terms of tracking performance while keeping respective qualities of the two other control laws. However, its computational complexity is too much for real-time applications and the non-convex optimization due to the non linearity leads to local mimima whose quality are difficult to evaluate.

In this paper, the MPC controller has been linearized around a reference trajectory [2]. This reference trajectory is time-variable and is the output of the local planification algorithm. Such a linearization allows to overcome the two common drawbacks of nonlinear MPC schemes (computational burden and non-convexity of the problem). The linear formulation presented here can run in real-time on commercial grade computers. To keep the same behaviour as the nonlinear MPC algorithm given in [7], a formulation has been derived to introduce a penalty on the control signal derivative.

The continuous model $\dot{X} = f(X)$ for the MPC algorithm is shown in equation (1). A second order model has been chosen for the yaw rate. This model represents the inner loop as designed in [7]. A first order model has been kept for the longitudinal control as it gives satisfactory accuracy in our application. The state is defined as $X = (x, y, \psi, r, v, \dot{r})$ and the input $U = (r_d, v_d)$. In this paper, r denotes a yaw rate, ψ a heading, v a linear speed.

$$f(X) = \begin{pmatrix} v\cos\psi \\ v\sin\psi \\ r \\ \dot{r} \\ \tau_v^{-1}(v_d - v) \\ -2\zeta_r\omega_r\dot{r} - \omega_r^2(r - G_r r_d) \end{pmatrix} \quad (1)$$

Where τ_v is the first order time constant of the longitudinal actuator model, ζ_r is the damping of the 2^{nd} order lateral actuator model, ω_r the pulsation and G_r the gain. The number of states is $n = 6$ and the number of inputs is $p = 2$. The model conventions are shown in Fig. 2.

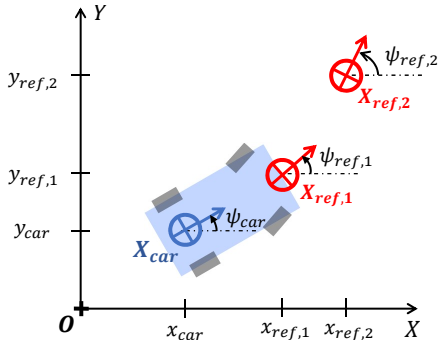

Fig. 2. Model conventions

The series of reference points $X_{ref,i}$ in Fig. 2 define a reference trajectory to track and is generated by a trajectory planner. Each $X_{ref,i}$ has the same dimension than the model state X.

A discrete model is derived from the continuous one with a first order Taylor series around the linearization point (X_{ref}, U_{ref}) with sampling time T_s:

$$\tilde{X}(k+1) = A_c(k, T_s)\tilde{X}(k) + B_c(k, T_s)\tilde{U}(k) \quad (2)$$

Where U_{ref} is a reference input that allows to follow perfectly the reference states $X_{ref,i}$. The difference vectors are defined as $\tilde{X} = X - X_{ref}$ and $\tilde{U} = U - U_{ref}$. The reference input U_{ref} is computed through an "inversion" of the model between successive reference states. For example, between two reference states $X_{ref,0}$ and $X_{ref,1}$, the reference input would be ideally defined by the perfect following equality:

$$X_{ref,1} = A_c(0, T_s)X_{ref,0} - B_c(0, T_s) * U_{ref} \quad (3)$$

As the input matrix $B_c(0, T_s)$ is not always invertible, the reference input has been computed by means of a least square approximation.

The equation (2) can be vectorized over the MPC prediction horizon of size N_{MPC}:

$$\tilde{\mathbf{X}}(k+1) = \mathbf{A_c}(k)\tilde{X}(k|k) + \mathbf{B}_c(k)\tilde{\mathbf{U}}(k) \quad (4)$$

Where $\tilde{\mathbf{X}}(k+1) = (\tilde{X}(k+1),\ \dots\ , \tilde{X}(k+N_{MPC}))^T$ and $\tilde{\mathbf{U}}(k) = (\tilde{U}(k),\ \dots\ , \tilde{U}(k+N_{MPC}-1))^T$

The computation of the model matrices $\mathbf{A_c}(k)$ and $\mathbf{B_c}(k)$ is explained in [2]. They depend on the non vectorized model matrices $A_c(k+i, T_s)$ and $B_c(k+i, T_s)$ at the different timesteps i in the MPC horizon. To mitigate the discretization errors without increasing the control sampling rate, the model matrices have been computed at an upsampled rate T_{up}.

The optimization problem can be expressed as the usual quadratic optimization problem for MPC [2] with the cost function $J(k)$ defined as:

$$J(k) = \frac{1}{2}\tilde{\mathbf{U}}(k)^T\mathbf{H}(k)\tilde{\mathbf{U}}(k) + \mathbf{f}^T\tilde{\mathbf{U}}(k) \tag{5}$$

With:

$$\begin{aligned}\mathbf{H}(k) &= 2\left(\mathbf{B_c}(k)^T\mathbf{Q}\mathbf{B_c}(k) + \mathbf{R}\right)\\ \mathbf{f}(k) &= 2\mathbf{B_c}(k)^T\mathbf{Q}\mathbf{A_c}(k)\tilde{X}(k)\end{aligned} \tag{6}$$

In this formulation, the current difference between the state and the first reference $\tilde{X}(k)$ needs to be fully measurable to propagate the model.

The matrix $\mathbf{Q} = \text{diag}(Q_1, ..., Q_{N_{MPC}})$ is used for weighting the penalty of the state error $\tilde{X}$ in the cost function. Each Q_i is a n-by-n square matrix containing the weights for each state. In this application, these weights have been kept constant over the MPC horizon. Thus, each Q_i is defined as:

$$Q_i = (Q_{11}, Q_{22}, \ ... \ , Q_{nn}) \tag{7}$$

The matrix $\mathbf{R}$ weights the error to the reference input series $\mathbf{U}_{ref}$. $\mathbf{R}$ is built the same way as the matrix $\mathbf{Q}$: it is a diagonal matrix of N_{MPC} square blocks $R_i = \text{diag}(R_{11}, R_{22})$.

An augmentation of the cost function is proposed in this paper to include terms on the control signal derivatives (which could easily be translated into cost on the states derivatives). This term is often used in nonlinear MPC algorithms to smooth the control signal. In this application, it is also used as a way to make the MPC algorithm less sensitive to noise in the reference trajectory. Such a noisy reference can happen because of noisy localization data or when following a vehicle.

In this formulation, the derivatives are computed by means of finite differences. Let $\mathbf{U}$ be the input series to impose the weight on, $\mathbf{U}_{ref}$ the reference input and $\tilde{\mathbf{U}} = \mathbf{U} - \mathbf{U}_{ref}$.

The subscript Δ denotes backwards differences operation. Hence $\mathbf{U}_\Delta$ is the vector of backwards differences of the terms in $\mathbf{U}$.

For vectorized notations, the subscript 0 denotes the terms for the first $N_{MPC} - 1$ timesteps in an input vector. The subscript 1 denotes the terms for the last $N_{MPC} - 1$ timesteps. Hence, $\mathbf{U}_\Delta = \mathbf{U}_1 - \mathbf{U}_0$.

The aim is to find two matrices $\mathbf{H}_d$ and $\mathbf{f}_d$ so that:

$$\|\mathbf{U}_\Delta\|^2 = \tilde{\mathbf{U}}^T\mathbf{H}_d\tilde{\mathbf{U}} + \mathbf{f}_d^T\tilde{\mathbf{U}} \tag{8}$$

As a consequence, the matrices $\mathbf{H}_d$ and $\mathbf{f}_d$ (after weighting) can be added to $\mathbf{H}(k)$ and $\mathbf{f}(k)$ in the quadratic problem.

Developing $\|\mathbf{U}_\Delta\|^2$ in equation (8) and identifying the quadratic and linear terms yields the following definition for $\mathbf{H}_d$:

$$\mathbf{H}_d = \begin{pmatrix} \mathbf{I_p} & -\mathbf{I_p} & & & \mathbf{0} \\ -\mathbf{I_p} & 2\mathbf{I_p} & -\mathbf{I_p} & & \\ & \ddots & \ddots & \ddots & \\ & & -\mathbf{I_p} & 2\mathbf{I_p} & -\mathbf{I_p} \\ \mathbf{0} & & & -\mathbf{I_p} & \mathbf{I_p} \end{pmatrix} \tag{9}$$

Where $\mathbf{H}_d$ is a square matrix of size N_{MPC}-by-p (N_{MPC} is the size of the MPC horizon and p is the number of model inputs). The matrix $\mathbf{I_p}$ is the identity matrix of size p. The vector $\mathbf{f}_d$ is a column vector of dimension N_{MPC} defined as:

$$\mathbf{f}_d = 2\begin{pmatrix}\mathbf{U}_{ref,0} - \mathbf{U}_{ref,1} \\ 0\end{pmatrix} + 2\begin{pmatrix}0 \\ \mathbf{U}_{ref,1} - \mathbf{U}_{ref,0}\end{pmatrix} \tag{10}$$

With each block in the vector having p rows.

The new matrices for the optimization are finally obtained as follows:

$$\begin{aligned}\mathbf{H}(k) &= 2\left(\mathbf{B_c}(k)^T\mathbf{Q}\mathbf{B_c}(k) + \mathbf{R} + \mathbf{R}_d\mathbf{H}_d\right)\\ \mathbf{f}(k) &= 2\left(\mathbf{B_c}(k)^T\mathbf{Q}\mathbf{A_c}(k)\tilde{X}(k) + \mathbf{R}_d\mathbf{f}_d\right)\end{aligned}$$

Where $\mathbf{R}_d$ is used to weight the penalty on the control signal derivative. It is built in the same way as $\mathbf{R}$. The scalar weight for the lateral input derivative is denoted $R_{d,11}$ and for the longitudinal input $R_{d,22}$.

This formulation introduces a trade-off between the tracking performance and the passenger comfort in the usual MPC formulation (favorizing the smoothness of the control signal). Usually, the weight $\mathbf{R}$ is used to tune the convergence rate of MPC controllers. The main drawback is that it penalizes the difference between U and U_{ref}: this difference can be noisy due to the computation of U_{ref}. It arises because the model inversion that is used to get U_{ref} from X_{ref} tends to amplify noise on U_{ref}. If the aim is to be able to give coarse reference paths to the MPC and limit the sensitivity to model inaccuracies, the convergence should not be set by increasing $\mathbf{R}$.

Because of this new weight $\mathbf{R}_d$ on the control signal derivative in the penalty function, no hard constraints need to be introduced in the MPC optimization for comfort features. Instead, comfort requirements will be dealt with the trajectory planner, in order to keep even aggressive emergency trajectories controllable by the MPC algorithm.

However, a constraint on the maximal required curvature has to be implemented to consider the vehicle's geometry and mechanical constraints. With the proposed linear MPC formulation and choice of the states, it has to be implemented through an approximation. From the variables in $U = (r_d, v_d)^T$, the desired curvature is $c = r_d/v_d$ which is not a linear relation with respect to U. As a consequence, the curvature is expressed approximately with $\bar{c} = r_d/v_{ref}$ around the linearization trajectory. This approximation holds perfectly as long as the vehicle's speed is close to the reference speed.

Additionally, it has been chosen not to implement tracking constraints on the vehicle's state as they can destabilize the optimization if being impossible to fulfil. In that case, a

linear quadratic solver may prefer to violate constraints on the input (maximum speed or yaw rate) and give completely unacceptable solutions. A solution with a monitoring block has been preferred, in order to introduce a form of feedback on the tracking performance. This is detailed in the next section.

C. MPC behaviour monitoring

As tracking performance should not be compromised for safe trajectory tracking, a behaviour monitoring module for the MPC algorithm has been developed. This module uses the knowledge of the MPC controller performance to generate a risk estimation associated to what is currently asked to the MPC controller. It takes advantage of the predictive nature of the MPC. This information is intended to be used in a trajectory planner in order to find acceptable trajectories risk-wise.

The safety monitoring module has two subfunctions in order to assess:

- The risk of violating the comfort constraints
- The risk of lateral tracking constraint violation

In this application, comfort constraints are defined as state constraints on the yaw rate and yaw rate derivative. The risk of violating the comfort constraints is evaluated as a critical time $t_{c,comfort}$ at which those constraints would be violated by tracking the current trajectory. The optimal input found $\mathbf{U_{opt}}$ is used to propagate the states of the vehicle over the MPC horizon. The predicted states are then checked for violation of the yaw rate and yaw acceleration constraints. If such a point is found at the "critical" index $i_{c,comfort}$, then:

$$t_{c,comfort} = T_s i_{critical} \tag{11}$$

For tracking accuracy monitoring, an estimation of the uncertainty of the MPC predicted lateral error $\hat{e}_y$ is first carried out. At each time step k, the N_{MPC} past inputs are applied from the vehicle's position N_{MPC} time steps ago. The lateral error predictions $[\hat{e}_y(k-N_{MPC}), ..., \hat{e}_y(k)]$ obtained are compared with the known lateral errors $[e_y(k-N_{MPC}), ..., e_y(k)]$ up to the present time. At the time k, it gives the errors associated to the prediction of e_y:

$$\tilde{e}_y(i,k) = \hat{e}_y(k-i) - e_y(k-i) \ (\forall i \in [1, N_{MPC}]) \tag{12}$$

These errors are due to modelling inaccuracies and measurement errors. They are compiled in a matrix $\mathbf{E}_{y,model}$ of size (N_{MPC}, N_{mem}) over a finite number of timesteps N_{mem}. A column j of $\mathbf{E}_{y,model}$ contains the model errors for each step in the MPC horizon computed j timesteps ago.

The uncertainty associated to $\hat{e}_y$ is computed from the data in $\mathbf{E}_{y,model}$. Assuming an unbiased normal distribution for each $\hat{e}_y(i)$, its uncertainty is computed as the standard deviation (denoted $\sigma_{e_y}(i)$) of the prediction errors for each step i in the MPC horizon:

$$\sigma_{e_y}(i) = \sqrt{\frac{1}{N_{MPC}-1} \sum_{j=1}^{N_{MPC}} \mathbf{E}_{y,model}(i,j)^2} \tag{13}$$

The bigger the memory time N_{mem}, the finer the estimation of the standard deviation, but the more lag it has. This can be problematic if the model error is dependant on external parameters such as the curvature of the road. The main risk is to underestimate the standard deviation of $\hat{e}_y$, and as a consequence to have too much confidence in the model. A short horizon of $N_{mem} = 5$ has been chosen.

Thanks to the information on e_y uncertainty, a probabilistic risk prediction for the tracking error violation can be carried out. The result of this prediction will be a critical time $t_{c,\hat{e}_y}$ within the MPC horizon at which the probability of overshooting the lateral tracking constraint is above a given threshold probability P_c (typically 5%):

$$\begin{cases} i_{c,\hat{e}_y} = \min\limits_{i \in [1, N_{MPC}]} (i, \text{ so that } P\left(\hat{e}_y(i) \geq e_{y,max}\right) \geq P_c) \\ t_{c,\hat{e}_y} = T_s i_{c,\hat{e}_y} \end{cases} \tag{14}$$

It is equivalent as looking for the first i for which the required confidence interval of $\hat{e}_y$ is not included in $[-e_{y,max}, e_{y,max}]$.
For instance, the 95% confidence interval of $\hat{e}_y(i)$ ($P_c = 5\%$) is defined as:

$$I_{c,95\%} = [\hat{e}_y(i) - 2\sigma_{e_y}(i), \hat{e}_y(i) + 2\sigma_{e_y}(i)]$$

Thus, the index looked for is the first i so that:

$$|\hat{e}_y(i)| + 2\sigma_{e_y}(i) \geq e_{y,max} \tag{15}$$

This relation can be generalized for any threshold probability P_c, knowing the confidence intervals of the normal distribution of standard deviation $\sigma_{e_y}(i)$. This way of computing a critical time is more conservative than using $\hat{e}_y$ without considering its uncertainty. Its aim is to allow earlier notice of risky situations and avoid false negatives (non notification about a potentially risky situation).

The two indexes developed in this section provide in-depth and clearly understandable information about different kind of risks linked to the trajectory tracking performance. Their additional computational cost is very low because all the information used comes from the MPC algorithm, which makes them interesting as a systematic probabilistic risk-monitoring approach for MPC applications. The risk evaluation defined in this section will be assessed in section III-B.

III. EXPERIMENTS

The simulations shown in this section have been performed with the ROS framework and the 4D-Virtualiz simulation engine[1]. A screenshot of the environment is shown in Fig. 3. This simulation environment provides a realistic physical model for the vehicle and actuators as well as integration within the ROS framework. The vehicle used in simulation is an *IPCar*, an electric urban vehicle of maximal speed $3m/s$ and minimum turning radius $R_{min} \approx 3m$. The roads used in simulation are an exact copy of the *Plateforme d'Auvergne*

[1]http://www.4d-virtualiz.com/. A demonstration video has been joined to the paper submission.

Fig. 3. 4D-Virtualiz simulation environment

pour les Véhicules Intelligents[2] (PAVIN). The PAVIN is a half scale test track representing a neighbourhood, with a traffic lights regulated intersection, a roundabout and various urban elements such as stop signs, curbs, narrow roads and pedestrian crossings. Thus, realistic urban driving situations can be transposed in the PAVIN at lower speeds. All the vehicle's sensors have been modeled as on the real ones: Real-Time Kinematics GPS (RTK-GPS), LIDAR, odometry sensors on the back wheels, angle sensors on the front wheels. The yaw acceleration is not measured and thus is simply computed by means of finite differences.

For the experiments presented here, the GPS data is perfect and has been sampled at $F_s = 10Hz$. The odometry sensors on the back wheels give a noisy and low resolution estimation of the yaw rate and linear speed. No gyroscopes are available on these test vehicles and thus have not been added to the simulated vehicle.

A. MPC behaviour analysis

The behaviour of the MPC controller has been assessed against its main tuning coefficicients for lateral dynamics R_{11} and the term introduced in this paper $R_{d,11}$. The other tuning term for lateral dynamics Q_{11} has been kept constant, since multiplying all the coefficients by a scalar ultimately does not change the "shape" of the cost function. The aim of this section is to show that the adaptation of the weight on the control signal derivative to the linear case makes sense and allows to tune the MPC controller so that it gives a smooth trajectory even in the presence of noisy perception/reference trajectories. The path followed is a 90^o left hand turn of minimum radius of curvature $R_c \approx 4m$. Its characteristics are shown in shown in Fig. 4. Both path are discrete trajectories with a $0.3m$ spatial sampling. The blue one has been created with raw data from a vehicle trajectory, to simulated a mobile target tracking in multi-vehicle navigation scenarios. The orange path is a smooth version of the blue one, with a clean curvature curve as can be seen on the bottom plot. The tracking algorithm applies a local filtering to the path, and the bottom plot shows the

[2]IPDS, "http://ipds.univ-bpclermont.fr," Nov. 2017, The Institut Pascal Data Sets.

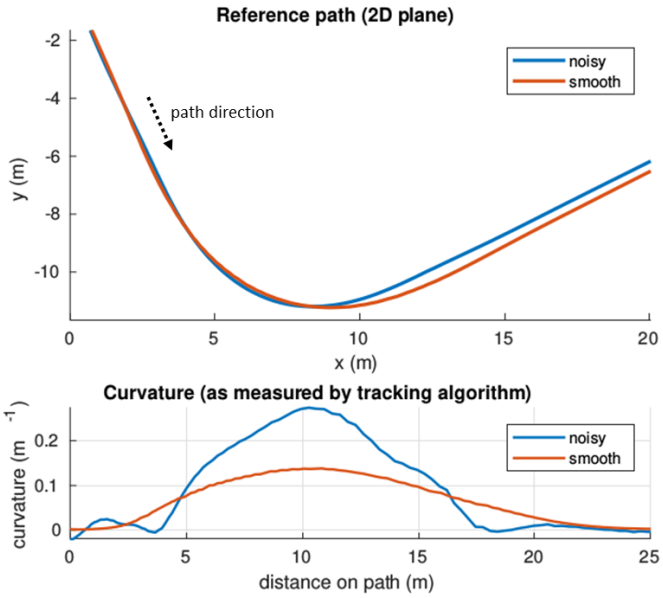

Fig. 4. Reference trajectory for MPC coefficients assessment

result of that. It can be noted that for two very different curvature profiles the $x-y$ path is almost identical. However the curvature is lined to the estimation of r_d in the tracking algorithm (the reference yaw rate) to which the actual yaw rate has to be close to. This noisy curvature will then impede the tracking performance of the MPC algorithm without the proposed weight on derivative.

In this section, the weights for the state error on the x-axis and the y-axis have been kept constant at $Q_{11} = Q_{22} = 1$. The reference trajectory is followed at a constant speed of $v_{ref} = 2$m/s. Thus, only the lateral control signal has been plot. For the plots, the measured yaw rate data has been smoothed *a posteriori* with a moving average of 5 samples.

The first experiment shows the effect of the introduced term $R_{d,11}$ when following a smooth reference trajectory. The second experiment shows the effect of $R_{d,11}$ when following a noisy reference trajectory. The default MPC controller parameters have been summarized in table I. This table also contains actuator parameters, which have been identified with a least-squares method with data from the simulator.

The MPC time horizon is $N/F_s = 1.5s$ and is long enough to contain the actuators dynamics. The loop rate F_s has been chosen so that the time horizon can be sufficiently long without a high number of samples N. The coefficients R_{11} and R_{22} have been found through automated testing of a range of values to ensure a smooth convergence without degradation of the tracking performance.

In Fig. 5, the curve for $R_{d,11} = 0$ corresponds to a classic linear MPC formulation. Increasing the coefficient $R_{d,11}$ shows very little improvement when following the smooth reference path. Even though the control signal r_d is smoother, the only noisy event at $t = 4s$ present in r_d in the classic case has no influence on the measured yaw rate

TABLE I

PARAMETER VALUES FOR MPC BEHAVIOUR ANALYSIS

parameter	**notation**	**value**
reference speed	v_{ref}	2 m/s
loop rate	F_s	10 Hz
MPC horizon	N	15
weight on longitudinal error	Q_{22}	1
default weight on lateral ref. input	R_{11}	3
weight on longitudinal ref input	R_{22}	5
lateral actuator gain	G_r	0.92
lateral actuator pulsation	ω_r	6.9 rad/s
lateral actuator damping	ζ_r	0.7
longitudinal actuator time constant	τ_v	0.5

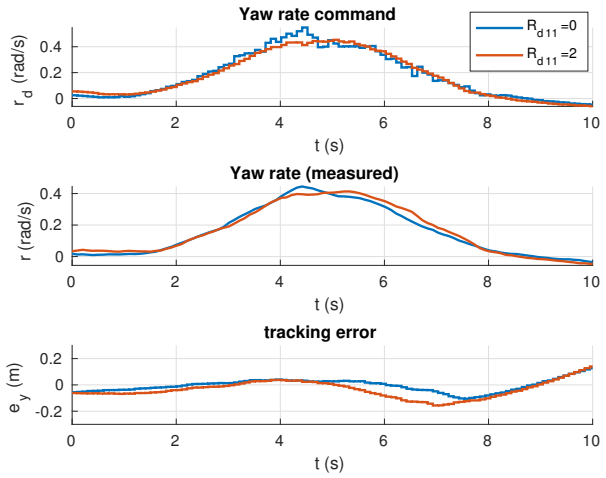

Fig. 5. Effect of $R_{d,11}$ coefficient, smooth reference trajectory

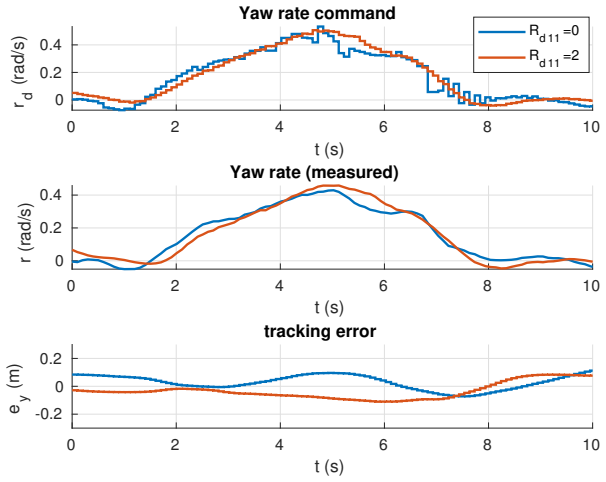

Fig. 6. Effect of $R_{d,11}$ coefficient, noisy reference trajectory

and does not introduce any oscillations.

However, the introduced weight on the control signal derivative shows a beneficial effect under a noisy reference trajectory in Fig. 6. It shows an improvement in the control signal smoothness and in the measured yaw rate, while not degrading the tracking performance. The artifacts on the command signal around $t = 5s$ and between $t = 7s$ and $t = 8s$ are filtered. The first consequence is that the actuator effort is reduced. The second one is that the yaw rate command with $R_{d,11} = 2$ is made closer to the shape of the smooth turn in 4. Note that as the turn is taken at constant speed, the curvature profile is equivalent to the yaw rate profile, to a multiplicative constant. It also helps preventing the noise on the control signal r_d that is picked by the classic formulation, and successfully smoothes the turn exit.

In conclusion, the weight on the input derivative allows to find a better tradeoff between tracking accuracy and control signal smoothness by acting as a filter for noisy reference signals and inaccuracies in the model. Ultimately, it makes the MPC controller more confortable. It also has the added benefit of not needing the higher order derivatives to be in the model. Those higher order state derivatives are often noisy and difficult to estimate.

B. Risk monitoring

For the experiments with the risk monitoring module, the noisy reference trajectory has been used.

A limit value of $e_{y,max}$ has been set for the lateral tracking error. A higher speed of $2.5m/s$ has also been used, which is close to the maximum speed of the *IPCar*.

Two simulations have been run, to show the interest of assessing the risk as presented in the article. In the first one, the reference speed of $2.5m/s$ is carried out through the whole maneuver, independantly of the risk prediction. In the second one, the reference speed has been set to go down to $2m/s$ as soon as $t_{c,\hat{e}_y} < 0.5s$. It is thus a very simple form of speed replanning depending on the risk estimation. Results are shown in Fig. 7.

Fig. 7 shows the results for the lateral tracking risk estimator. The blue line corresponds to the simulation where no speed replanning is applied, and the orange one to the simulation where the speed is reduced if $t_{c,\hat{e}_y} < 0.5s$. The top plot shows the critical time $t_{c,\hat{e}_y}$ based on a 5% risk of violating the lateral tracking constraint. At $t = 2.9s$, the replanning happens for the second simulation, as seen in the bottom plot. As a consequence, it effectively manages to prevent the overshooting of the lateral error constraint (dotted line on the middle plot). When no speed replanning happens, the tracking constraint ends up being violated. A more elaborate path planner could re-plan the speed in a smoother way, instead of the step that has been imposed on the reference speed in the second simulation.

IV. CONCLUSIONS

In this paper, a control architecture for risk and comfort management for an urban vehicle has been proposed. This architecture solves the problem of trajectory tracking for

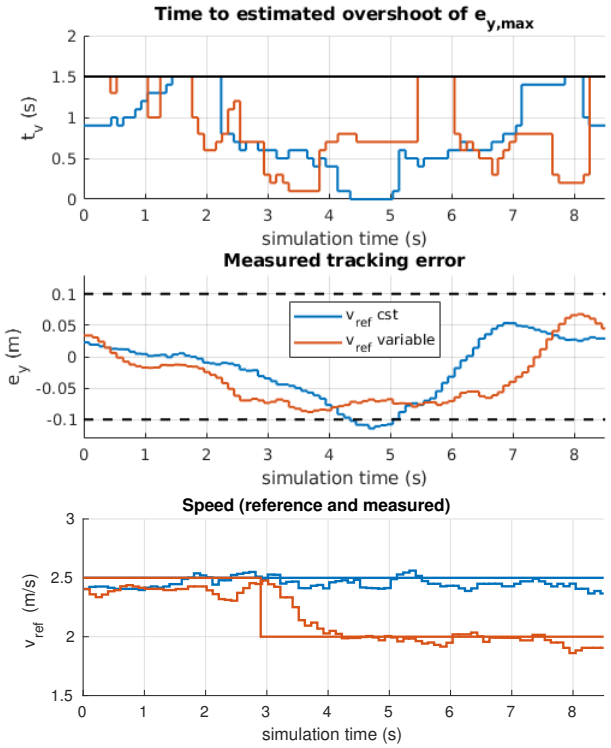

Fig. 7. Lateral tracking risk prediction and speed replanning

ground vehicles while being robust to noise on the reference trajectory. A safety monitoring module has been added in order to have a probabilistic way of assessing the risk linked to the trajectory tracking (defined as the violation of constraints). This safety management has been shown to be able to foresee future dangerous situations and prevent them.

Thanks to these developments, further work on planning and collaborative intersection management will be made easier. With the proposed MPC controller, such algorithms would not have to deal with smoothness of the trajectory because it is dealt with in the trajectory tracking algorithm. Thus the planning can have a lower complexity. The freed computing power can then be used for probabilistic risk-based approaches that are the continuity of the approach hereby developped. For instance, risk estimation could be used in platooning. When following a leader, the following vehicles could notify the leader if it is too risky for them to keep the formation.

ACKNOWLEDGMENT

This work has been sponsored by the French government research program Investissements d'avenir through the RobotEx Equipment of Excellence (ANR-10-EQPX-44) and the IMoBS^3 Laboratory of Excellence (ANR-10-LABX-16-01), by the European Union through the program Regional competitiveness and employment 2014-2020 (FEDER - AURA region) and by the AURA region.

REFERENCES

[1] Lounis Adouane. *Autonomous Vehicle Navigation From Behavioral to Hybrid Multi-Controller Architectures*. Taylor & Francis CRC Press, 2016.

[2] F Künhe, J Gomes, and W Fetter. "Mobile Robot Trajectory Tracking Using Model Predictive Control". In: *II IEEE latin-american robotics* (2005), pp. 1–7.

[3] Guilherme V. Raffo, Guilherme K. Gomes, Julio E. Normey-Rico, Christian R. Kelber, and Leandro B. Becker. "A predictive controller for autonomous vehicle path tracking". In: *IEEE Transactions on Intelligent Transportation Systems* 10.1 (Mar. 2009), pp. 92–102.

[4] Benjamin Gutjahr, Lutz Groll, and Moritz Werling. "Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC". In: *IEEE Transactions on Intelligent Transportation Systems* (2016), pp. 1–10.

[5] Tianyu Gu and John M. Dolan. "Toward human-like motion planning in urban environments". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, June 2014, pp. 350–355.

[6] Jean Baptiste Braconnier, Roland Lenain, and Benoit Thuilot. "Ensuring path tracking stability of mobile robots in harsh conditions: An adaptive and predictive velocity control". In: *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, May 2014, pp. 5268–5273.

[7] Charles Philippe, Lounis Adouane, Benoit Thuilot, Antonios Tsourdos, and Hyo Sang Shin. "Risk and comfort management for multi-vehicle navigation using a flexible and robust cascade control architecture". In: *2017 European Conference on Mobile Robots, ECMR 2017*. IEEE, Sept. 2017, pp. 1–7.

[8] Jesús Morales, Jorge L. Martínez, María A Martínez, and Anthony Mandow. "Pure-pursuit reactive path tracking for nonholonomic mobile robots with a 2D laser scanner". In: *Eurasip Journal on Advances in Signal Processing* 2009.1 (2009), p. 935237.

[9] José Miguel Vilca, Lounis Adouane, and Youcef Mezouar. "A novel safe and flexible control strategy based on target reaching for the navigation of urban vehicles". In: *Robotics and Autonomous Systems* 70 (Aug. 2015), pp. 215–226.