Real-Time Constrained Trajectory Planning and Vehicle Control for Proactive Autonomous Driving with Road Users

Ivo Batkovic[1,2], Mario Zanon[3], Mohammad Ali[2], and Paolo Falcone[1]

***Abstract*— For motion planning and control of autonomous vehicles to be proactive and safe, pedestrians' and other road users' motions must be considered. In this paper, we present a vehicle motion planning and control framework, based on Model Predictive Control, accounting for moving obstacles. Measured pedestrian states are fed into a prediction layer which translates each pedestrians' predicted motion into constraints for the MPC problem.**

Simulations and experimental validation were performed with simulated crossing pedestrians to show the performance of the framework. Experimental results show that the controller is stable even under significant input delays, while still maintaining very low computational times. In addition, real pedestrian data was used to further validate the developed framework in simulations.

I. INTRODUCTION

Since a decade, autonomous driving technologies have emerged with the potential for increased safe and efficient driving [1]. While one can expect autonomous driving technologies to be deployed first in structured environments such as highway driving and low-speed parking [2], other scenarios, such as urban driving, arguably pose a greater challenge due to the presence of non-autonomous road users, e.g. pedestrians, cyclists, and other vehicles. Hence, the research focus needs to be directed to deriving models for predicting the stochastic behavior of human road users, while also avoiding collisions with them.

This paper targets the autonomous driving problem in complex urban environments where road users are present. The autonomous vehicle needs to drive along a pre-defined route, while also ensuring that collisions are avoided. Therefore, it is necessary that the vehicle is provided with information about the environment, such as where a road user will most likely be in the future, in order to be proactive and plan for a collision-free path. Therefore, the vehicle motion planning and control needs to be handled in real-time, hence a low computational complexity is needed.

While safety is the dominating factor when designing driving trajectories for autonomous driving, passengers' comfort also needs to be considered. The main contributing factors to ride discomfort are considered to be high levels of jerk and acceleration [3]. Several methods have been proposed where these factors are taken in consideration and optimized for driving scenarios on highways or country roads [4], [5]. Similarly, the work in [6], [7] minimizes jerk profiles under fixed travel times set beforehand.

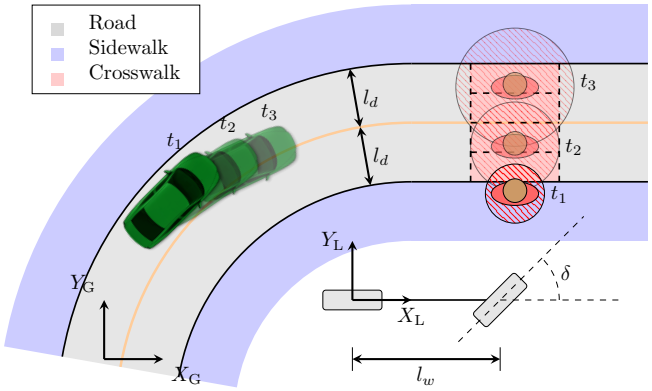

Fig. 1. An illustration of a path planning problem. The vehicle needs to plan a trajectory in time subject to constraints such as road boundaries, but also moving pedestrians (red circles). The kinematic bicycle model is depicted in the lower part of the figure.

For urban driving, [8] presents a framework for trajectory planning by decomposing the problem in spatial path planning and velocity profile planning. However, the planner does not consider reactive behaviors such as responding to dynamic objects in the environment. Other frameworks [9], [10] use grid or graph-based methods to either plan paths around pedestrians or stop for them, while making an assumption that alternative paths exist, which may not always hold true in urban scenarios. Planners using human comfort as optimization are also able to plan paths around obstacles [11], [12], but avoid stopping as it is not always optimal. Similar to other planners, [13] uses a path-velocity decomposition to generate path and velocity profiles in parallel. While the planner is able to slow down, or come to a complete stop for pedestrians, it does not consider pedestrian movements in time, which might affect the planning capability.

In this paper we provide a general real-time framework to handle autonomous driving in urban scenarios with pedestrian crossing intersections. Unlike the common approach in literature, we do not decouple the longitudinal and lateral control of the vehicle. Instead, we solve the trajectory planning and vehicle control problem simultaneously, with the assumption that a higher level navigation layer provides a nominal driving route, e.g. center of the road lane, that the vehicle needs to track. Model Predictive Control (MPC) is used to generate optimal trajectories, while considering

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation, and by the COPPLAR project (VINNOVA. V.P. Grant No. 2015-04849).

[1] Ivo Batkovic and Paolo Falcone are with the Mechatronics group at the Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden {ivo.batkovic,falcone}@chalmers.se

[2] Ivo Batkovic, and Mohammad Ali are with the research department at Zenuity AB {ivo.batkovic,mohammad.ali}@zenuity.com

[3] Mario Zanon is with the IMT School for Advanced Studies Lucca mario.zanon@imtlucca.it

other road users by predicting their motion in time. When considering other road users, other approaches in the literature often predict the future motion with primitive methods, e.g. constant velocity or acceleration models. In this paper we use an environment-aware predictor, developed in [14], to obtain more accurate future predictions, and unify the collision avoidance problem simultaneously with the vehicle motion and control problem.

We ensure collision avoidance by transforming the predicted motion and uncertainty of other road users into constraints within the MPC formulation. Doing so, allows us to plan a trajectory in time, while being proactive and avoiding collisions with moving road users or other static obstacles. Fig. 1 shows an example of how predicted pedestrian positions can be used for collision avoidance. The red circles express regions at different planning times that the vehicle is not allowed to enter. It is important to note that we aim at addressing a nominal behavior which guarantees safety and comfort, but which also requires an emergency layer which would intervene in case something unexpected happens. The safety layer is the subject of ongoing research.

This paper is structured as follows. In Section II we introduce the problem formulation and our framework along with the used vehicle and pedestrian prediction models. The framework is evaluated, both in simulations and real experiments, for scenarios with simulated pedestrians moving near an intersection in Section III. Finally, we draw conclusions and outline future research directions in Section IV.

II. Problem Formulation and Framework

An autonomous driving application needs above all to be safe, and in order safely coexist with a higher level of riding comfort, the vehicle behavior needs to be proactive. By predicting the future evolution of the driving environment, one can incorporate cautiousness to minimize the risk of collisions with surrounding traffic participants (pedestrians, cyclists, cars). In the following, we first introduce the system architecture, then we explain the vehicle and pedestrian models used in our decision problem. Finally, we present the reference path generation, cost function, and system constraints used to formulate our control objectives.

A. System architecture

At each sampling instant, we assume that the vehicle receives estimated vehicle states $\hat{\mathbf{x}}_0$ from an estimator, previous guesses of state and control input trajectories $\mathbf{x}_k^{\mathrm{g}}$ and $\mathbf{u}_k^{\mathrm{g}}$ respectively, and a path reference ξ^{c} and $\mathbf{v}^{\mathrm{c}}$ from a nominal path module. State measurements $\mathbf{x}_k^{\mathrm{meas}}$ of a detected pedestrian[1] are sent to a module that generates predictions $\mathbf{x}_k^{\mathrm{ped}}$ of the pedestrian positions in time. The planner uses the estimated states, the state and control guesses, the path references, and the pedestrian predictions to control the vehicle along the path, while avoiding collisions. This paper focuses on the planner represented as the largest dashed rectangle in Fig. 2.

[1]Although our approach can be extended to other road users, for convenience of exposition we'll refer to pedestrians

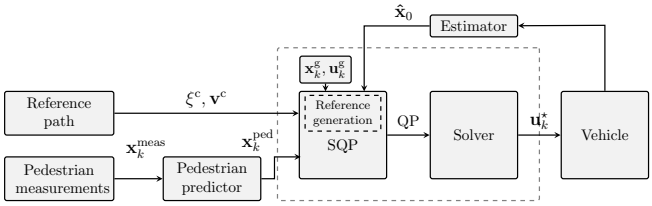

Fig. 2. Representation of the closed loop architecture.

B. Vehicle model

For urban autonomous driving, lateral, longitudinal, and yaw dynamics are quite fast [15]. Therefore, a kinematic bicycle model was chosen for its simplicity and comparable accuracy with a dynamic one [16] in the scenarios considered in this paper. A sketch of the vehicle model is shown in Fig. 1.

We model the vehicle motion with states $\mathbf{x} \in \mathbb{R}^6$, and controls $\mathbf{u} \in \mathbb{R}^2$, defined as

$$\mathbf{x} := \begin{bmatrix} x & y & v & \theta & \delta & \omega \end{bmatrix}^\top, \quad \mathbf{u} := \begin{bmatrix} a & \delta_{\mathrm{sp}} \end{bmatrix}^\top,$$

where we denote the position coordinates of the rear wheel in the absolute reference frame as x, y, the velocity in direction θ as v, and the steering angle and steering angle rate as δ and ω, respectively. The acceleration and steering angle setpoint are denoted respectively by a and δ_{sp}, where the setpoint refers to desired steering angle at the wheel base.

With the chosen state representation, the dynamics are described by the bicycle equations as

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \\ \dot{\theta} \\ \dot{\delta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ a \\ v/l_w \tan\delta \\ \omega \\ w_0^2(\delta_{\mathrm{sp}} - \delta) - 2\zeta\omega \end{bmatrix} =: f(\mathbf{x}, \mathbf{u}), \quad (1)$$

where the only model parameters are: the wheel base l_w and the parameters w_0 and ζ, introduced to describe a second order model of the steering wheel actuator.

For simplicity, we chose to omit the longitudinal jerk in the model. It could, however, easily be implemented without any greater complication or performance impact.

C. Pedestrian model

The performance of a planner is predominantly limited by the accuracy of the information it is provided with: generating plans with highly inaccurate information about the surrounding environment will result in suboptimal and possibly unsafe or conservative plans. Since our planner solves an optimal control problem over a prediction horizon, a description of the environment, e.g. pedestrian positions and road boundaries, are necessary throughout the planning horizon.

We chose to use the flexible and fairly accurate method presented in [14] to generate pedestrian predictions needed for our planner. A simple representation of the road geometry is used in order to assign possible paths (hereafter referred to as references), e.g. sidewalks and crosswalks, where a pedestrian might walk. The pedestrian motion is modeled

using a unicycle model, which is assumed to follow the reference, thanks to a closed-loop regulator. Process noise is added to the model, and state uncertainty is predicted by propagating its covariance. Using the closed-loop system model, pedestrian trajectories and state covariances can be predicted along the road geometry.

Moreover, if a pedestrian is walking on a road that has a bifurcation, e.g. the road splits into multiple walking paths, the prediction method takes into account all possible directions and generates predictions along these paths.

D. Problem statement

The objective of the vehicle is to safely follow any given path as close as possible, while driving comfortably and satisfying constraints arising from the physical limitations, but also other road users, e.g pedestrians, cyclists and other vehicles.

This problem can be formulated as a finite horizon, constrained optimal control problem. However, since the vehicle model and constraints are nonlinear, we are faced with the following Nonlinear Model Predictive Control (NMPC) problem

$$\min_{\mathbf{x},\mathbf{u}} \sum_{k=0}^{N-1} \begin{bmatrix} \mathbf{x}_k - \mathbf{r}_k^{\mathbf{x}} \\ \mathbf{u}_k - \mathbf{r}_k^{\mathbf{u}} \end{bmatrix}^\top W_k \begin{bmatrix} \mathbf{x}_k - \mathbf{r}_k^{\mathbf{x}} \\ \mathbf{u}_k - \mathbf{r}_k^{\mathbf{u}} \end{bmatrix} \tag{2a}$$
$$+ \begin{bmatrix} \mathbf{x}_N - \mathbf{r}_N^{\mathbf{x}} \end{bmatrix}^\top W_N \begin{bmatrix} \mathbf{x}_N - \mathbf{r}_N^{\mathbf{x}} \end{bmatrix}$$
$$\text{s.t.} \quad \mathbf{x}_0 = \hat{\mathbf{x}}_0, \tag{2b}$$
$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \tag{2c}$$
$$h(\mathbf{x}_k, \mathbf{u}_k) \leq 0, \tag{2d}$$

where N is the prediction horizon, W_k is the stage cost matrix, $\mathbf{x}_k$ and $\mathbf{u}_k$ are the state and control input, $\mathbf{r}_k^{\mathbf{x}}$ and $\mathbf{r}_k^{\mathbf{u}}$ are the state and control input reference, constraint (2b) enforces that the prediction starts at the current state, constraint (2c) enforces the system dynamics, and constraint (2d) enforces constraints such as, e.g. actuator limits and obstacle avoidance.

With the algorithms described in [17] the Nonlinear Program (NLP) (2) can be rewritten using a Sequential Quadratic Programming (SQP) approach, which sequentially approximates the NLP with a Quadratic Program (QP) (3). Following the SQP approach we linearize the system model with a previous guess $(\mathbf{x}_k^{\mathrm{g}}, \mathbf{u}_k^{\mathrm{g}})$ and use any numerical discretization scheme to obtain the sensitivities

$$A_k = \frac{\partial f}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_k^{\mathrm{g}}, \mathbf{u}_k^{\mathrm{g}}}, B_k = \frac{\partial f}{\partial \mathbf{u}}\bigg|_{\mathbf{x}_k^{\mathrm{g}}, \mathbf{u}_k^{\mathrm{g}}}, b_k = f(\mathbf{x}_k^{\mathrm{g}}, \mathbf{u}_k^{\mathrm{g}}) - \mathbf{x}_{k+1}^{\mathrm{g}},$$

for every prediction time step k. The constraints C_k, D_k, $\mathbf{d}_k^{\mathrm{l}}$, $\mathbf{d}_k^{\mathrm{u}}$ C_N, $\mathbf{d}_N^{\mathrm{l}}$, and $\mathbf{d}_N^{\mathrm{u}}$ can be computed in a similar manner from the constraints (2d). With the sensitivities, the problem is then formulated into a QP and solved, where the solution is used to update the previous guess. This process is either iterated until convergence, or only done once, thus following the real time iteration (RTI) scheme [18]. The SQP approach transforms the problem into the following QP

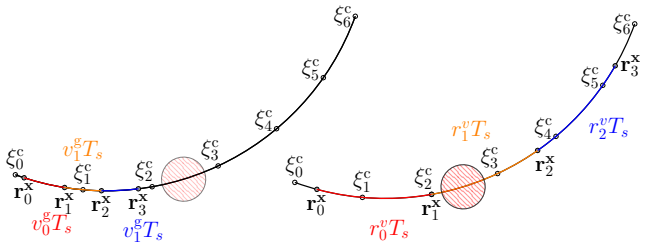

Fig. 3. Illustration of generated reference points $\mathbf{r}_k^{\mathbf{x}}$. The left and right curves show the reference points obtained by integration with either the previous velocity profile v_k^{g} or the velocity reference r_k^v. Using the velocity reference r_k^v, the generated reference points $\mathbf{r}_k^{\mathbf{x}}$ does not account for the obstacle (red circle).

$$\min_{\mathbf{x},\mathbf{u}} \sum_{k=0}^{N-1} \begin{bmatrix} \mathbf{x}_k - \mathbf{r}_k^{\mathbf{x}} \\ \mathbf{u}_k - \mathbf{r}_k^{\mathbf{u}} \end{bmatrix}^\top \begin{bmatrix} Q_k & S_k \\ S_k^\top & R_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_k - \mathbf{r}_k^{\mathbf{x}} \\ \mathbf{u}_k - \mathbf{r}_k^{\mathbf{u}} \end{bmatrix} \tag{3a}$$
$$+ \begin{bmatrix} \mathbf{x}_N - \mathbf{r}_N^{\mathbf{x}} \end{bmatrix}^\top Q_N \begin{bmatrix} \mathbf{x}_N - \mathbf{r}_N^{\mathbf{x}} \end{bmatrix}$$
$$\text{s.t.} \quad \mathbf{x}_0 = \hat{\mathbf{x}}_0, \tag{3b}$$
$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k + b_k, \tag{3c}$$
$$\mathbf{d}_k^{\mathrm{l}} \leq C_k \mathbf{x}_k + D_k \mathbf{u}_k \leq \mathbf{d}_k^{\mathrm{u}} \tag{3d}$$
$$\mathbf{d}_N^{\mathrm{l}} \leq C_N \mathbf{x}_N \leq \mathbf{d}_N^{\mathrm{u}}. \tag{3e}$$

The tuning parameters here are the stage cost matrices Q_k, R_k and S_k, and the prediction horizon N. Note that all matrices in this problem formulation are in general time-varying.

At every sampling instant, Problem (3) is solved, and only the first control input $\mathbf{u}_0$ is applied to the system. However, the solutions $\mathbf{x}$ and $\mathbf{u}$ are used to update the guesses $(\mathbf{x}_k^{\mathrm{g}}, \mathbf{u}_k^{\mathrm{g}}) = (\mathbf{x}_{k+1}, \mathbf{u}_{k+1})$, where k spans the prediction horizon.

E. Reference generation

The generation of the trajectory reference deserves a specific discussion, since it is a crucial component of the control scheme and, as such, has a strong effect on the closed-loop behavior. In standard MPC implementations, the reference is updated from one time step to the next by shifting: $(\mathbf{r}_k^{\mathbf{x}}, \mathbf{r}_k^{\mathbf{u}}) = (\mathbf{r}_{k+1}^{\mathbf{x}}, \mathbf{r}_{k+1}^{\mathbf{u}})$. In our setup instead, we recompute the reference at every time instant in order to avoid aggressive controller behaviors when the vehicle is forced to slow down due to the presence of an obstacle.

In particular, rather than relying on a reference trajectory, we choose a reference path and a reference velocity along it. The advantage of such choice is that, if the vehicle is forced to slow down by the presence of an obstacle, the controller will not try to recover the lost time and instead it will simply try to bring the vehicle back at the desired velocity. In our research, we adopt an approach which is very similar to the one proposed in [19] to address these issues. Differently from [19], we eliminate the curvilinear path parameter σ from the problem formulation and we relate $\dot{\sigma}$ to the vehicle's velocity directly.

We assume that a reference velocity $v^{\mathrm{c}}(\sigma)$ and a curve $\xi(\sigma) = (x^{\mathrm{c}}(\sigma), y^{\mathrm{c}}(\sigma), \theta^{\mathrm{c}}(\sigma), \delta^{\mathrm{c}}(\sigma), \omega^{\mathrm{c}}(\sigma), a^{\mathrm{c}}(\sigma), \delta_{\mathrm{sp}}^{\mathrm{c}}(\sigma))$ parametrized in the curvilinear coordinate σ are available.

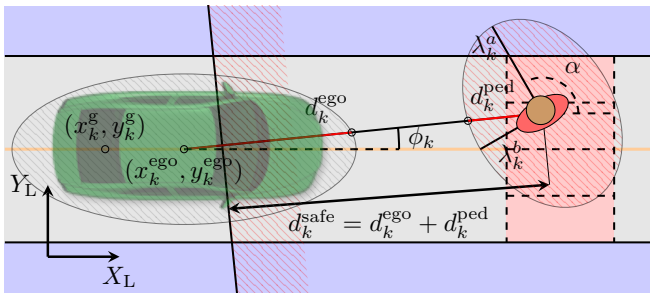

Fig. 4. Illustration of the regions occupied by the vehicle and pedestrian. A linear constraint is expressed by the black line, distancing itself with distance d_k^{safe} from the pedestrian.

Using the current position (x_0, y_0), we define the initial reference coordinate σ_0 by projecting (x_0, y_0) onto the curve $\xi(\sigma)$ such that $\sigma_0 = \arg\min_\sigma \|(x_0, y_0) - (x^{\text{c}}(\sigma), y^{\text{c}}(\sigma))\|_2$. Then, we define reference points at time kT_{s} by means of $\sigma_k = \sigma(kT_{\text{s}})$. In order to obtain a meaningful reference in the presence of obstacles, we propose to use

$$\sigma_k = \int_0^t (1 - \kappa(\tau)e_y(\tau))^{-1} v(\tau)\cos(e_\psi(\tau))d\tau, \quad (4)$$

where κ, e_y, and e_ψ is the curvature, lateral error and orientation error w.r.t to the path. This guarantees that the predicted reference accommodates for the presence of obstacles [20]. Using the reference velocity in the computation of $\sigma(t)$, instead, would ignore the presence of obstacles and create the potential for unstable behavior, especially in the presence of sharp turns. In the context of the RTI scheme, $\sigma(t)$ is obtained by integration of the velocity computed at the previous time instant.

In practice, the curves $\xi(\sigma)$ and $v(\sigma)$ might not be available and the reference generation layer might have access only to a set of data points ξ_p^{c} and v_p^{c}. Then, a curve can be obtained by interpolation. In this paper, we use a simple piecewise linear interpolation and the formula $\sigma_{k+1} = \sigma_k + v_k^{\text{g}}\cos(e_{\psi_k})T_{\text{s}}$ for approximate integration. By selecting σ_k according to v_k^{g} rather than r_k^v, the algorithm is able to account for the presence of obstacles, as explained in Fig. 3. Note that in the second case, the orientation reference will refer to a point on the path far from the previously predicted vehicle position and, therefore, will result in an incorrect orientation reference and unwanted driving behavior.

Moreover, we assume that the only available data concerns positions and velocities, such that we need to define suitable references for the other states and controls. The position references are obtained as $(r_k^x, r_k^y) = (x^c(\sigma_k), y^c(\sigma_k))$, while the heading reference r_k^θ is computed by

$$r_k^\theta = \tan^{-1}\left(\frac{r_{k+1}^y - r_k^y}{r_{k+1}^x - r_k^x}\right), \quad (5)$$

and the steering angle and steering angle rate references are set as the initial angle δ_0 and zero, respectively, across all prediction times k. The reference for the control input is set as zero for the acceleration, and δ_0 for the steering setpoint.

F. Cost function

Often, matrices Q_k, R_k, and S_k are chosen to have the diagonal form

$$\begin{aligned} Q_k &= \text{diag}(q_1, ..., q_m), \\ R_k &= \text{diag}(r_1, ..., r_n), \quad S = 0, \end{aligned} \quad (6)$$

where m and n denote the state and control input dimensions.

As mentioned in Section II-E, in the presence of obstacles, trying to force the system to catch up with a predefined evolution of the curvilinear coordinate σ can yield an undesirably aggressive behavior. Therefore, we propose to remove the term penalizing this deviation from the cost. We do so by only penalizing the lateral deviation from the path ξ^{c} through the term

$$\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}^\top \begin{bmatrix} \cos(-r_k^\theta) & -\sin(-r_k^\theta) \\ \sin(-r_k^\theta) & \cos(-r_k^\theta) \end{bmatrix} \begin{bmatrix} x_k - r_k^x \\ y_k - r_k^y \end{bmatrix}\right)^2.$$

Then, matrix Q_k becomes

$$Q_k = \text{blockdiag}(q_1 T(-r_k^\theta),\ q_2,\ q_3,\ q_4,\ q_5), \quad (7)$$

with

$$T(\theta) = \begin{bmatrix} \sin^2\theta & \cos\theta\sin\theta \\ \cos\theta\sin\theta & \cos^2\theta \end{bmatrix}. \quad (8)$$

With this cost formulation we avoid undesirably aggressive behavior and we still ensure that the cost matrices are positive semi-definite.

G. System constraints

The states and control inputs are subject to a set of constraints, which concerns mainly safety aspects, comfort and physical limitations of the vehicle, which are formulated in (3d)-(3e). These consist of the road boundary limits, but also of occupied areas of predicted pedestrians' positions.

1) Road boundary constraints: In order to decide on the lateral bounds of the road, we use the previous guesses $\mathbf{x}^{\text{g}}$ to project the positions $(x_k^{\text{g}}, y_k^{\text{g}})$ onto the path. Knowing the projection point $(\hat{x}_k^{\text{g}}, \hat{y}_k^{\text{g}})$, and the orientation $\hat{\theta}_k^{\text{g}}$, it is possible to express a lateral bound l_d on the path as

$$\Delta^{\text{road}} - l_d \le x_k\sin(-\hat{\theta}_k^{\text{g}}) + y_k\cos(-\hat{\theta}_k^{\text{g}}) \le \Delta^{\text{road}} + l_d, \quad (9)$$

where $\Delta^{\text{road}} = \hat{x}_k^{\text{g}}\sin(-\hat{\theta}_k^{\text{g}}) + \hat{y}_k^{\text{g}}\cos(-\hat{\theta}_k^{\text{g}})$.

2) Pedestrian constraints: Information about future pedestrian states is provided by a prediction layer outside of the MPC framework. The predictions consist of positions $(x_k^{\text{ped}}, y_k^{\text{ped}})$, the major and minor axes λ_k^a and λ_k^b corresponding to the covariance ellipse of a Gaussian probability density function, and the orientation α in a global frame. With this information, occupied pedestrian regions are defined in the time domain. Similarly, we define a safety ellipse around the vehicle that describes the space it occupies. An illustration is provided in Fig. 4.

The covariance ellipses are linearized using the previous vehicle positions $(x_k^{\text{g}}, y_k^{\text{g}})$ and heading θ_k^{g} to find the vehicle center $(x_k^{\text{ego}}, y_k^{\text{ego}})$. The predicted pedestrian positions $(x_k^{\text{ped}}, y_k^{\text{ped}})$ and the vehicle center are used to compute the safety distances $d_k^{\text{ego}}$and d_k^{ped} for the vehicle and pedestrian

using the relative orientation ϕ_k. Considering the two safety distances, we express a linear constraint around the rear wheel of the vehicle as

$$\Delta^{\text{ped}}\mathbf{l}_b \leq \Delta^{\text{ped}} \begin{bmatrix} x_k \\ y_k \end{bmatrix}, \tag{10}$$

where $\Delta^{\text{ped}} = -\begin{bmatrix} \cos\phi_k & \sin\phi_k \end{bmatrix}$ and

$$\mathbf{l}_b = \begin{bmatrix} x_k^{\text{ped}} - d_k^{\text{safe}}\cos\phi_k - (x_k^c - x_k^{\text{guess}}) \\ y_k^{\text{ped}} - d_k^{\text{safe}}\sin\phi_k - (y_k^c - y_k^{\text{guess}}) \end{bmatrix}. \tag{11}$$

The resulting linear constraint, together with a physical illustration of the parameters, is illustrated in Fig. 4.

III. Simulations & Experimental Validation

In this section we present results from a traffic scenario simulation with a crossing pedestrian, and the ego vehicle. To model the pedestrian, we used a trajectory from the data set in [14]. To further verify the framework, we also provide experimental results for a traffic intersection with a simulated pedestrian.

A. Simulation Setup

The controller was first tested in a simulated intersection scenario with a real pedestrian trajectory taken from [14]. In this scenario, the ego vehicle is traveling along a straight road towards an intersection which a pedestrian is approaching, see Fig. 5.

We generated sensitivites A_k and B_k and b_k with a fourth order Runge-Kutta method, while ensuring that 5 integrator steps was sufficiently accurate, with a discretization time of $\Delta t = 0.05\text{s}$, and the following parameter values

$$l_w = 2.984\text{m}, \quad w_0 = 20\text{s}^{-1}, \quad \zeta = 0.9\text{s}^{-1}. \tag{12}$$

The estimated state $\hat{\mathbf{x}}_0$ was obtained by integration, using an integrator with error control to guarantee the accuracy of the solution. The vehicle controller used

$$\begin{aligned} Q_k &= \text{blockdiag}(2T(-r_k^\theta),\ 0.1,\ 10,\ 0.1,\ 10), \\ R_k &= \text{blockdiag}(2, 1), \qquad S_k = 0,\ N = 100, \end{aligned} \tag{13}$$

and the state and control inputs were subject to the following constraints

$$\begin{aligned} -1 \leq v_k \leq 20,\ |\delta_k| \leq 0.4942,\ |\omega_k| \leq 0.1765, \\ -2 \leq a_k \leq 1,\ |\delta_{k,\text{sp}}| \leq 0.4942, \end{aligned} \tag{14}$$

where all values are given in standard SI-units.

Finally, the lateral bound on the road boundary constraint (9) was set to be $l_d = 1\text{m}$. In order to ensure feasibility of the optimization problem (3), we relax constraint (9) with an L1-penalty. This will ensure that if there exists a feasible solution for problem (3), then the relaxed problem yields the same solution [21, Prop. 6].

B. Experimental Setup

The closed loop controller was experimentally validated at the Astazero[2] test-track outside Gothenburg, Sweden. The experiment consisted of a traffic scenario including our ego vehicle, and a simulated walking pedestrian. A simulated pedestrian was used due to safety reasons. The vehicle needed to make a 90-degree left turn in the intersection, while a simulated pedestrian was moving close-by and could possibly cross. Fig. 7 shows the considered scenario, where the vehicle needs to drive along the orange line that represents the center of the driving lane, with an approaching pedestrian from the right.

1) Test vehicle: The vehicle used in the experiment was a Volvo XC90 T6 petrol-turbo SUV with the real-time open source software OpenDLV[3] that interfaced sensors and actuators to an external computer for control. A Laptop computer (i7 2.8GHz, 16GB RAM) running Ubuntu 16.04 as a Virtual Machine was used to receive sensor data, compute the control signals, and send actuation requests to the vehicle through OpenDLV through an ethernet connection. The vehicle was equipped with a Real Time Kinematic (RTK) GPS receiver for high accuracy positioning, and the estimated state $\hat{\mathbf{x}}_0$ was obtained with an Extended Kalman Filter (EKF).

2) Experimental Details: The interface between OpenDLV and the vehicle exhibited a delay of roughly 300ms when sending signals to the steering actuator, but also reading from it. Therefore the state space vector in (1) was augmented with additional time-delayed states for the steering angle to model the input delay. Since we estimated the steering wheel actuator dynamics, dead reckoning was used for estimating the steering wheel angle and steering wheel angle rate. Furthermore, due to safety features in the vehicle, the actuator limited the steering wheel angle in a speed-dependent way while in autonomous driving mode. To model this, we implemented the linear constraint

$$a_1 v_k + b_1 \delta_k \leq c_1, \quad a_2 v_k + b_2 \delta_k \geq c_2. \tag{15}$$

The sensitivities were generated in the same way as for the simulation, and used same parameter values as in (12) and tuning parameters as in (13).

C. Results

We implemented the proposed framework in MATLAB first for the simulations, and then auto generated the code into C++, where it was interfaced with OpenDLV for experimental testing. Both implementations used the open source software ACADO [22] for auto generation of the sensitivities, and the solver from HPMPC [23] to solve the QP problem with the RTI scheme.

1) Simulation: Fig. 5 illustrates the open-loop solutions across different time instants. The MPC controller is generating control inputs to minimize the deviation from the reference path and reference velocity $r_k^v = 10\text{m/s}$. For time $t = 1.25\text{s}$ the open-loop solution is not affected

[2]See http://www.astazero.com/ for more information

[3]See https://opendlv.org/ for more information.

by the predicted pedestrian positions, and the controller is generating inputs just to stay on the path. At time $t = 7.5$s the vehicle has adapted its speed and managed to plan a path around the pedestrian instead of coming to a full stop. For times $t = 10$s and $t = 12.5$s, we can see how the vehicle finally passes the pedestrian, and the final closed-loop trajectory. The corresponding closed-loop states and control inputs are presented in Fig. 6.

The aggressive behavior on the steering rate is due to the deviation of the actual pedestrian position from the predicted one. When the vehicle is close to the pedestrian, the control authority is reduced and such uncertainties result in large control actions. Future research will investigate approaches to mitigate these undesirable behaviors.

Lastly, the runtime of the framework in MATLAB resulted in an average solution time of 22.4ms with a standard deviation of 2.3ms, and longest timing of 34.1ms.

2) Experiment: Fig. 8 shows the open-loop solutions across different time instants for the experiments at the test-track. The same controller from the simulation is used with the reference velocity $r_k^v = 5\text{m/s}$. For time $t = 13.75$s the open-loop solution is not affected by the pedestrian constraints, and the controller is generating inputs only to stay on the path. For time $t = 20$s the prediction algorithm propagates pedestrian positions through the intersection, where one of the predictions is crossing the vehicle's path. Since the predicted positions are transformed into constraints, the vehicle is forced to slow down and make sure that the pedestrian passes. This can subsequently be seen in times $t = 23.75$s through $t = 30$s. Fig. 9 shows the corresponding closed-loop states and control inputs.

The computer running the controller sent actuation signals at a rate of 20Hz. The actual runtime of the framework, including the pedestrian prediction method, resulted in an average solution time of 10.3ms, with a standard deviation of 2ms, and longest timing of 19.7ms. Thus, the runtime leaves a great margin to further increase the algorithmic complexity, or the sampling rate of the control scheme. In addition, since the controller ran in a Virtual Machine and not on the native computer operating system, there is a possibility to further reduce the computational time.

IV. Conclusions

In this paper we considered autonomous driving in urban environments with pedestrian crossings. We propose a general framework that solves the trajectory planning, and the longitudinal an lateral vehicle control problem both in simulations and real experiments. In addition, by combining predictions of the environment, it is shown that behaviors, such as slowing down, or stopping for a crossing pedestrian, are naturally included without any noteworthy increase in complexity. As shown, the framework already presents real-time performance, without being highly optimized, except for the choice of sensitivity generation through ACADO and the solver provided by HPMPC.

Future work will aim to research situations where pedestrians suddenly appear to the framework due to sensor occlusion. In addition, we will aim to further verify the framework in more complex intersections with crossing pedestrians. Finally, future research will aim at developing a control scheme with recursive feasibility guarantees, which is robust to unexpected events.

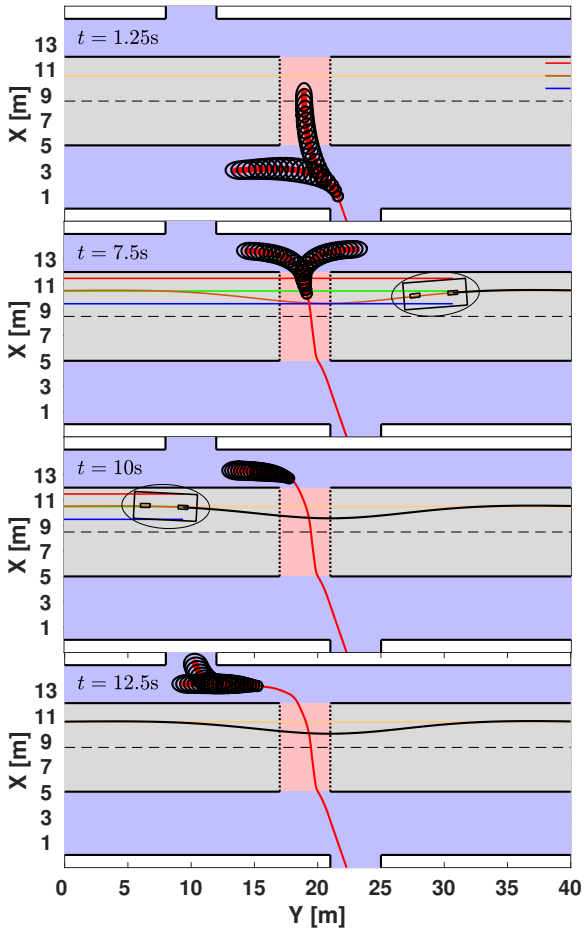

Fig. 5. Scenario with real pedestrian measurements. The blue and red lines are the road boundary constraints from (9), while the green line and brown lines are the projected reference (r_k^x, r_k^y) and and open-loop solution $\mathbf{x}$. The predicted pedestrian states are depicted as red points, and the uncertainties as black ellipses. The trailing black and red lines show the traveled path of the vehicle and pedestrian.

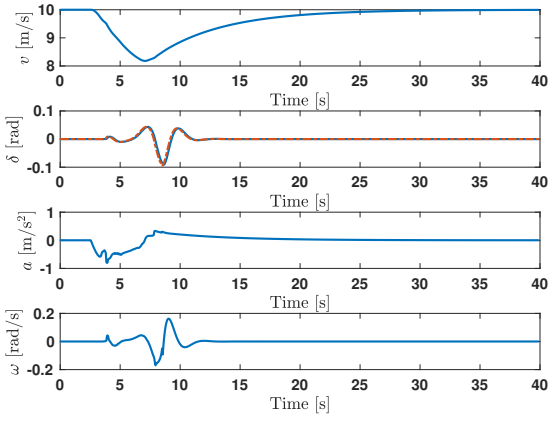

Fig. 6. Closed loop states from Fig. 5. The first plot shows the velocity profile and the second plot shows the steering angle (blue line), and steering angle setpoint (dashed brown line). The last two plots show the acceleration and steering angle rate.

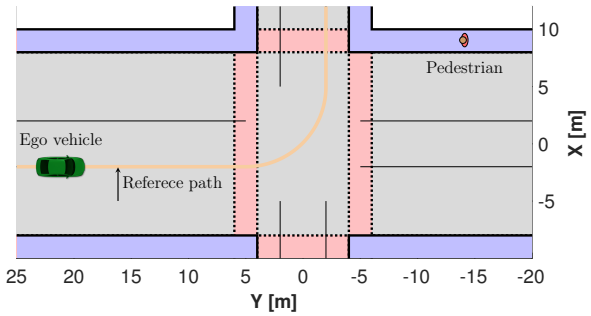

Fig. 7. The vehicle needs to drive along its given path, and is faced with a pedestrian moving towards the intersection, not knowing if the pedestrian will cross or not.

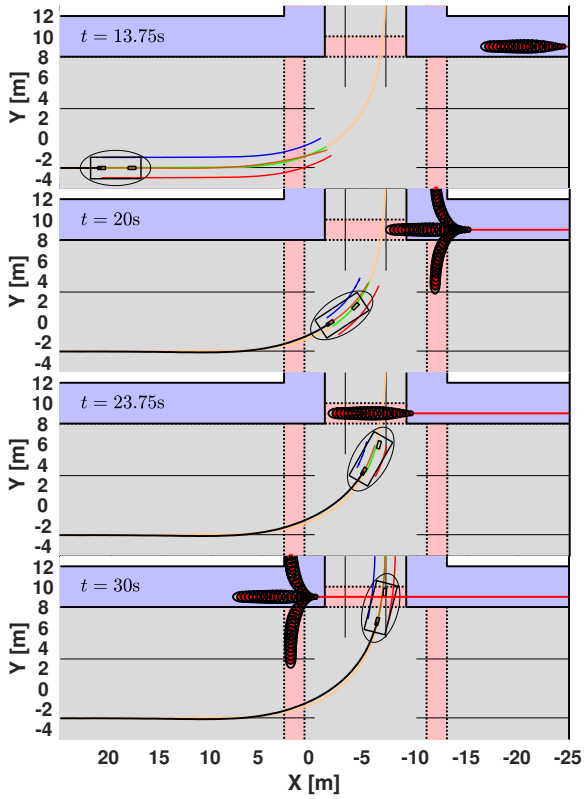

Fig. 8. Open-loop solutions across different time instants. The color scheme matches the one of Fig. 5.

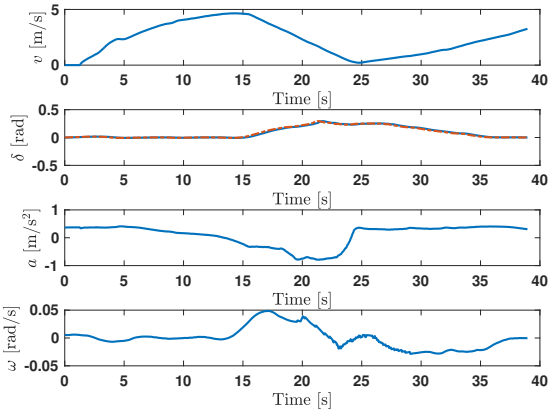

Fig. 9. Closed loop states from Fig. 8. The plots match the same states as those of Fig. 6.

REFERENCES

[1] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, *et al.*, "Making bertha drive - an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[2] J. Becker, M.-B. A. Colas, S. Nordbruch, and M. Fausten, "Boschs vision and roadmap toward fully autonomous driving," in *Road vehicle automation*. Springer, 2014, pp. 49–59.

[3] P. Nilsson, L. Laine, and B. Jacobson, "Performance characteristics for automated driving of long heavy vehicle combinations evaluated in motion simulator," in *IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 362–369.

[4] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for berthaa local, continuous method," in *IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 450–457.

[5] J. Nilsson, M. Brännström, E. Coelingh, and J. Fredriksson, "Longitudinal and lateral control for automated lane change maneuvers," in *IEEE American Control Conference (ACC)*, 2015, pp. 1399–1404.

[6] C. G. L. Bianco and M. Romano, "Optimal velocity planning for autonomous vehicles considering curvature constraints," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 2706–2711.

[7] C. G. L. Bianco, "Minimum-jerk velocity planning for mobile robot applications," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1317–1326, 2013.

[8] X. Li, Z. Sun, Z. He, Q. Zhu, and D. Liu, "A practical trajectory planning framework for autonomous ground vehicles driving in urban environments," in *IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1160–1166.

[9] R. Benenson, S. Petti, T. Fraichard, and M. Parent, "Integrating perception and planning for autonomous navigation of urban vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 98–104.

[10] J. Johnson and K. Hauser, "Optimal longitudinal control planning with moving obstacles," in *IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 605–611.

[11] J. Villagra, V. Milanés, J. P. Rastelli, J. Godoy, and E. Onieva, "Path and speed planning for smooth autonomous navigation," in *IEEE Intelligent Vehicles Symposium (IV)*, 2012.

[12] J. Villagra, V. Milanés, J. Pérez, and J. Godoy, "Smooth path and speed planning for an automated public transport vehicle," *Robotics and Autonomous Systems*, vol. 60, no. 2, pp. 252–265, 2012.

[13] R. G. Cofield and R. Gupta, "Reactive trajectory planning and tracking for pedestrian-aware autonomous driving in urban environments," in *IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 747–754.

[14] I. Batkovic, M. Zanon, N. Lubbe, and P. Falcone, "A Computationally Efficient Model for Pedestrian Motion Prediction," *ArXiv e-prints*, Mar. 2018, available at https://arxiv.org/abs/1803.04702.

[15] R. Rajamani, "Lateral and longitudinal tire forces," in *Vehicle Dynamics and Control*, 2nd ed. Springer, 2012, ch. 13, pp. 355–395.

[16] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1094–1099.

[17] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear mpc: bridging the gap via the real-time iteration," *International Journal of Control*, pp. 1–19, 2016.

[18] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on control and optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[19] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," in *Proceedings of the 48th IEEE Conference on Decision and Control. CDC, 2009.* IEEE, 2009, pp. 8642–8647.

[20] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, "Time-optimal race car driving using an online exact hessian based nonlinear mpc algorithm," in *European Control Conference (ECC)*. EUCA, 2016, pp. 141–147.

[21] R. Hult, M. Zanon, S. Gros, and P. Falcone, "Optimal coordination of automated vehicles at intersections: Theory and experiments."

[22] R. Quirynen, S. Gros, B. Houska, and M. Diehl, "Lifted collocation integrators for direct optimal control in acado toolkit," *Mathematical Programming Computation*, vol. 9, no. 4, pp. 527–571, 2017.

[23] G. Frison, H. B. Sørensen, B. Dammann, and J. B. Jørgensen, "High-performance small-scale solvers for linear model predictive control," in *European Control Conference (ECC)*. EUCA, 2014, pp. 128–133.