# Motion Planning for Urban Autonomous Driving using Bézier Curves and MPC

Xiangjun Qian[1] Iñaki Navarro[2] Arnaud de La Fortelle[1] Fabien Moutarde[1]

***Abstract*— This paper presents a real-time motion planning scheme for urban autonomous driving that will be deployed as a basis for cooperative maneuvers defined in the European project AutoNet2030. We use a path-velocity decomposition approach to separate the motion planning problem into a path planning problem and a velocity planning problem. The path planner first generates a collision-free piecewise linear path and then uses quintic Bézier curves to smooth the path with $C^2$ continuity. A derive-free optimization technique Subplex is used to further smooth the curvature of the path in a best-effort basis. The velocity planner generates an optimal velocity profile along the reference path using Model Predictive Control (MPC), taking into account user preferences and cooperative maneuver requirements. Simulation results are presented to validate the approach, with special focus on the flexibility, cooperative-awareness and efficiency of the algorithms.**

## I. Introduction

### A. Motivation

Autonomous driving is becoming a promising technology to enhance traffic efficiency and reduce resources wasted on accidents and congestions. It is predicted that, by the date of 2030, more than 50% of vehicles on the road will be automated [1].

This work is supported by the European project AutoNet2030 [1] that shall develop multi-vehicle cooperative algorithms for various driving scenarios, such as platooning, convoy (multi-lane platoons) and lane change. This requires to develop a unified motion planning framework that can be tuned to cope with different individual driving scenarios and various cooperative maneuvers (flexibility and cooperative awareness). Moreover, the motion planner will run alongside other computationally demanding modules such as cooperative sensing/communication modules. Thus we need the motion planner to be fast within an embedded architecture (efficiency). The concerned vehicle platforms are illustrated in Fig. 1.

In the following, we survey existing literature and present our contribution.

[1]Xiangjun Qian, Arnaud de La Fortelle and Fabien Moutarde are with MINES ParisTech, PSL - Research University, Centre for Robotics, 60 Bd St Michel, 75006 Paris, France {xiangjun.qian, arnaud.de_la_fortelle, fabien.moutarde}@mines-paristech.fr

[2]Iñaki Navarro is with the Distributed Intelligent Systems and Algorithms Laboratory, School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne inaki.navarro@epfl.ch

This research was supported by the European FP7 project AutoNet2030 (Grant Agreement NO. 610542).


(a) (b)

Fig. 1: (a) Cybus platform of INRIA and (b) Automated truck of Scania

### B. Related work

Years of research have borne fruits to many motion planning frameworks [2]–[7]. The literature can be roughly divided into three categories of approaches: sampling based approaches, Model Predictive Control (MPC) based approaches and path-velocity decomposition approaches.

Sampling based approaches [2]–[4] sample directly the state space of the autonomous vehicle to obtain a set of feasible trajectories, and then select the best one to execute, subject to some cost function. Deterministic sampling approaches [2] discretize the state space using a conformal spatiotemporal lattice. Graph search methods are then adopted to find the optimal trajectory. The disadvantage of deterministic sampling is that it is not complete. In order to mitigate this problem, stochastic sampling approaches [3], [4] are proposed using Rapid Exploring Random Tree Star (RRT*) and its variants. RRT* is powerful in exploring the state space and is asymptotically complete. However, the resulted trajectory is usually jerky and needs further smoothing.

Sampling based approaches spend a large amount of computation resources to generate a large set of trajectories, while most of them are discarded. Model Predictive Control (MPC) based methods [8], [9] formulate the trajectory generation problem as an optimization problem over the state space and use gradient-descending techniques to find the optimal trajectory. The optimization problem is solved in a periodic fashion with a limited horizon to take into account new environmental information. This category of approaches is inherently more efficient than the sampling based one as the optimal trajectory can be found within only a couple of iterations with the help of the gradient information. Furthermore, MPC-based approach permits high-precision planning, for example, keeping a precisely given distance from its preceding vehicle, thanks to its optimization nature; while sampling based approaches cannot achieve the same level of accuracy unless a prohibitive

amount of samples is evaluated. However, a major drawback is that most optimization algorithms require constraints to be differentiable, which is often not the case for on-road obstacles. Moreover, the computation time may significantly increase if the environment is complex, *e.g.*, the presence of multiple obstacles.

Considering the specificities of on-road motion planning where the environment is highly structured, the path-velocity decomposition technique [10] can be adopted to break down the state space into two subspaces with lower dimensions thus drastically decreasing the computational complexity. The path planning problem determines a kinematically feasible (curvature-continuous) path along the road. Various path generation methods are proposed using cubic curvature polynomials [11], Bézier curves [5], [12], clothoid tentacles [13], Dubin's paths [14], and nonlinear optimization technique [7]. Quintic Bézier curves [5] is a promising approach because it is easy to compute and tune. However, further adaptation is necessary to make it compatible with real-time on-road autonomous driving. The velocity planning problem generates a speed profile that is feasible to the previously generated path. Previous work [7], [15], [16] usually assumes the velocity profile to have certain shape (polynomial, trapezoidal, *etc.*) and samples some terminal states to generate a set of profiles. A best profile is then selected using some cost function. However, a velocity profile with a priori form might not be optimal.

One drawback of the classical path-velocity decomposition lies in the handling of dynamic obstacles. For dynamic objects like the vehicles on the same direction or objects traversing the road, simply regulating the vehicle speed is sufficient for collision avoidance, while for other kinds of moving objects like a bicyclist moving at the border of the lane, lateral swerve is necessary. A precise solution of such scenario requires the unification of the path planning and the velocity planning. However, in urban scenario, objects that requires lateral swerve usually move at low speeds. We can approximate the slowly moving obstacle by a sweep volume up to certain temporal horizon so that it can be considered during the path planning stage.

### C. Contribution

This paper proposes a framework to satisfy the requirements of flexibility, cooperative-awareness and efficiency. We adopt a modified path-velocity decomposition approach. The path planner generates first a piecewise linear path that satisfies a tunable cost function and avoid static/slow objects on the road. Quintic Bézier curves are then used to smooth the path with $C^2$ continuity. The path is then further optimized using Subplex, a derive-free optimization technique. The design of the velocity planner is inspired by the MPC literature. We use an MPC controller to generate an optimal reference velocity profile. The MPC controller allows us to take into account various criteria (dynamic obstacle avoidance, comfort, fuel consumption, *etc.*). It permits high-precision velocity planning which is a major enabler of cooperative maneuvers. Finally, our design inherits the efficiency property of the classical path-velocity decomposition methods.

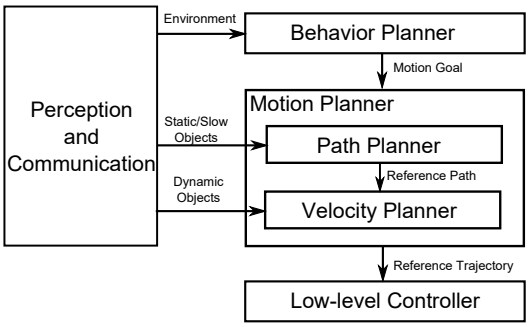


Fig. 2: Autonomous driving architecture: a simplified view

The rest of the paper is organized as follows. In Section II, the necessary background information is given. The path planner algorithm is described in Section III, while the velocity planner is presented in Section IV. The experimental results carried out in simulation are shown in Section V. Finally, Section VI draws the conclusions of this work.

## II. Preliminaries

### A. Architecture Overview

In Fig 2, we present a multi-level modular framework of our autonomous driving architecture. At the top is the behavior planner that is responsible for high-level decision making by fusing the data from sensors (laser scanners, radars, *etc.*), V2X devices, pre-calculated routes and traffic rules. The output of the behavior planner is an object called *motion goal* that contains information on the desired behavior of the vehicle for the next period of time. In this paper, we assume that the motion goal is available to the motion planner, and it carries information like user preferences, desired maneuvers and cooperative rules.

The motion planner adopts the proposed path-velocity decomposition approach to generate reference trajectories, which are in turn fed to the low-level controller for tracking.

As the focus of this paper is on the motion planner, we make the following assumptions. We assume that the perception and communication are reliable. The vehicle is convolved to the map thus we only need to consider the planning for a point. Finally, we assume that the map is preprocessed to provide a drivable region and a reference centerline.

### B. Vehicle Model

For high-level motion planning, we describe the vehicle using a particle motion model on a 2-D Cartesian plane:

$$\begin{aligned} \dot{x}(t) &= v(t)\cos(\phi(t)), \\ \dot{y}(t) &= v(t)\sin(\phi(t)), \\ \dot{\phi}(t) &= v(t)\kappa(t), \end{aligned} \tag{1}$$

where $\binom{x}{y}, v, \phi, \kappa$ are respectively 2-D position, velocity, heading and curvature. The curvature is related to the steering angle $\delta$ by

$$\kappa = \frac{\tan(\delta)}{L}, \tag{2}$$

where $L$ is the wheel base.

In the path-velocity decomposition approach, the path is firstly computed and then a velocity profile is used to parameterize the path. Assuming the path is given, (1) is decomposed into a function of the curvilinear abscissa $s$ (also called vehicle offset or station) and the velocity

$$\begin{aligned} \dot{x}(s) &= v(s)\cos(\phi(s)), \\ \dot{y}(s) &= v(s)\sin(\phi(s)). \end{aligned} \tag{3}$$

Finally the dynamics of the vehicle on the reference path is reduced to

$$\begin{aligned} \dot{s}(t) &= v(t), \\ \dot{v}(t) &= u(t), \end{aligned} \tag{4}$$

where $u \in [u_{min}, u_{max}]$ is the longitudinal acceleration. The lateral acceleration is given by $a_l(t) = v^2(t)\kappa(t)$ and should stay bounded: $a_l \in [-a_{l,max}, a_{l,max}]$.

### C. Quintic Bézier Curves

We briefly introduce quintic Bézier curves because of their importance hereafter. A quintic Bézier curve on a 2-D Cartesian plane is defined in parametric form as

$$B(\tau) = \sum_{i=0}^{5} \binom{5}{i} (1-\tau)^{5-i}\tau^i \mathbf{b}_i, \tau \in [0,1], \tag{5}$$

where $\mathbf{b}_{\{0\ldots5\}}$ are six control points. The curve satisfies the following properties

*1) End Point Interpolation:*

$$B(0) = \mathbf{b}_0, \ B(1) = \mathbf{b}_5. \tag{6}$$

*2) Tangent at End Points:*

$$B'(0) = 5(\mathbf{b}_1 - \mathbf{b}_0), \ B'(1) = 5(\mathbf{b}_4 - \mathbf{b}_5). \tag{7}$$

*3) Second Derivative at End Points:*

$$B''(0) = 20(\mathbf{b}_2 + \mathbf{b}_0 - 2\mathbf{b}_1), \ B''(1) = 20(\mathbf{b}_3 + \mathbf{b}_5 - 2\mathbf{b}_4). \tag{8}$$

*4) Curvature:*

$$k(\tau) = \frac{|B'(\tau) \times B''(\tau)|}{||B'(\tau)||^3}, \quad \tau \in [0,1]. \tag{9}$$

The properties above imply that $\mathbf{b}_{\{0\ldots5\}}$ are uniquely defined by $[B(0), B(1), B'(0), B'(1), B''(0), B''(1)]$.

## III. Path Planner

The path planner generates a curvature-continuous reference path subject to the directives of the behavior planner and conforms to user preferences. The path planning process has two stages: piecewise linear path generation and quintic Bézier curve interpolation, respectively discussed in the following. At the end of this section, we discuss the initialization and replanning scheme for the planner.

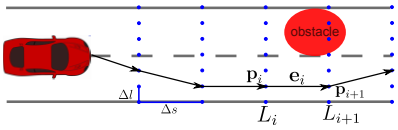



Fig. 3: Lane-adapted grid and piece-wise linear path.

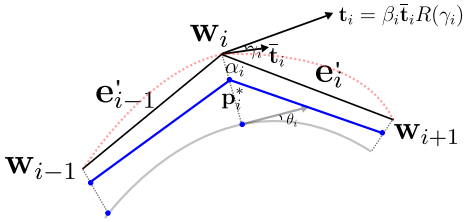



Fig. 4: Illustration of the interpolation process using quintic Bézier curves. The blue line is the first computed piecewise linear path. The red dashed line is the smoothed path using quintic Bézier curves.

### A. Piecewise Linear Path Generation

We adopt a classic technique to lay out a lane-adapted grid [2], with $N$ longitudinal layers separated by an interval $\Delta s$ and $K$ lateral samples with an interval $\Delta l$ for each layer (see Fig. 3). A piece-wise linear path can then be constructed by connecting sample points in a forward manner. Let $\mathbf{p}_i$ denote a sample point at layer $L_i$. Let $l_i$ be the lateral deviation of $\mathbf{p}_i$ from the centerline. Edge $\mathbf{e}_i$ connects a point $\mathbf{p}_i$ at layer $L_i$ to a point $\mathbf{p}_{i+1}$ at layer $L_{i+1}$. We adopt the dynamic programming technique to decide the optimal piecewise linear path. Different cost functions are defined for different actions. The optimal piecewise linear path represented by waypoints $\{\mathbf{p}^*_{i+1}\}$ is said to be optimal for the lane-keeping action such that:

$$\{\mathbf{p}^*_i\} = \arg \min_{\{\mathbf{p}_i \in L_i\}} \sum_{i=0}^{N-1} w_1 l_i^2 + w_2 ||\mathbf{e}_i||^2 + c_{obs}(\mathbf{e}_i) \tag{10}$$

where $l_i^2$ penalizes lateral deviation; $||\mathbf{e}_i||^2$ penalizes long edges; $c_{obs}(\mathbf{e}_i)$ equals to zero if edge $\mathbf{e}_i$ does not intersect with any obstacle (and the sweep volume of the obstacle) and equals to infinity otherwise. The parameters $w_1$ and $w_2$ can be adapted according to user preferences.

A different optimal criteria can be defined for the lane change action:

$$\{\mathbf{p}^*_i\} = \arg \min_{\{\mathbf{p}_i \in L_i\}} \sum_{i=0}^{N-1} w_1 \left(\frac{l_{i+1} - l_i}{\Delta t_p}\right)^2 + w_2 l_i^2 + c_{obs}(\mathbf{e}_i) \tag{11}$$

subject to

$$\frac{l_{i+1} + l_{i-1} - 2l_i}{\Delta t_p^2} < u_{lat} \tag{12}$$

The term $\left(\frac{l_{i+1}-l_i}{\Delta t_p}\right)^2$ penalizes the lateral speed with $\Delta t_p$ as the estimated time to traverse one layer. The constraint given by (12) is used to limit the lateral acceleration. Other terms are similar to the lane-keeping scenario.

Note that the programming problem is of polynomial complexity $O(NK^2)$, thus it can be solved in real-time. The proposed formulation is quite flexible: by using different

optimization criteria, we can generate different forms of paths suitable for different actions.

### *B. Interpolation and Optimization using Quintic Bézier Curves*

The piecewise linear path cannot be followed by autonomous vehicle. From Section II-B we see that curvature continuity ($C^2$ continuity for the curve) is the minimal requirement for a feasible path. We adopt quintic Bézier curves to achieve this goal. The approach is modified from [5] to consider a different cost function and more path tuning parameters. We also use a different optimization method.

We consider an arbitrary intermediate layer $L_i$. Let $B_{i-1}$ and $B_i$ be two quintic Bézier curves joining at point $\mathbf{w}_i$ at layer $L_i$. Let $\mathbf{t}_i$ and $\mathbf{a}_i$ be the tangent and second derivative vectors at $\mathbf{w}_i$. In order to have $C^2$ continuity, we need (see Fig. 4)

$$\begin{aligned} \mathbf{w}_i &= B_{i-1}(1) = B_i(0) \\ \mathbf{t}_i &= B'_{i-1}(1) = B'_i(0) \\ \mathbf{a}_i &= B''_{i-1}(1) = B''_i(0) \end{aligned} \tag{13}$$

The point $\mathbf{w}_i$ is determined by laterally shifting the optimal waypoint $\mathbf{p}_i^*$ obtained from the previous section with a small free parameter $\alpha_i$, such that

$$\mathbf{w}_i = \mathbf{p}_i^* + \alpha_i \left[ \begin{smallmatrix} \sin\theta_i \\ \cos\theta_i \end{smallmatrix} \right], \tag{14}$$

where $\theta_i$ is the angle of the corresponding centerline.

The tangent $\mathbf{t}_i$ is obtained by manipulating an initial guess $\bar{\mathbf{t}}_i$ as following:

$$\mathbf{t}_i = \beta_i R(\gamma_i)\bar{\mathbf{t}}_i. \tag{15}$$

The initial guess $\bar{\mathbf{t}}_i$ uses a widely accepted heuristic that sets the tangent perpendicular to the angular bisector of the neighboring line segments, such that $\bar{\mathbf{t}}_i = \frac{\mathbf{e}'_{i-1}}{||\mathbf{e}'_{i-1}||} + \frac{\mathbf{e}'_i}{||\mathbf{e}'_i||}$, where $\mathbf{e}'_i = \overrightarrow{\mathbf{w}_{i-1}\mathbf{w}_i}$ is the line segment connecting two waypoints. Note that $\mathbf{e}'_i$ is different from $\mathbf{e}_i$. The free parameter $\beta_i$ adjusts the magnitude of the tangent. The term $R(\gamma_i)$ rotates the tangent with a small parameter $\gamma_i$.

Given $\mathbf{w}_i$ and $\mathbf{t}_i$ for all $i \in 0, ..., N-1$, Piecewise *cubic* Bézier curves are fully determined but with only $C^1$ continuity at join points. As the curvature of *cubic* Bézier curve is in general small [5], we adopt the same heuristics as in [5] to set the second derivatives of quintic Bézier curves at a join point as a weighted mean of second derivatives of the corresponding *cubic* Bézier curves at this point. The interpolation process is illustrated in Fig. 4.

For the last layer $L_{N-1}$, the end point tangent is set as $\mathbf{t}_{N-1} = \beta_{N-1}R(\gamma_{N-1})\bar{\mathbf{t}}_{N-1}$, where $\bar{\mathbf{t}}_{N-1} = \frac{\mathbf{e}'_{N-2}}{||\mathbf{e}'_{N-2}||}$. $\mathbf{a}_{N-1}$ is simply set as the the second derivatives of the corresponding *cubic* Bézier curve at this point.

With the formulation above, the path constructed from Bézier curve is controlled by $3(N-1)$ free parameters $\{\alpha_i, \beta_i, \gamma_i\}_{i=\{1,...,N-1\}}$ to be optimized. The goal of optimization is to minimize the mean square curvature of the entire path given as

$$\min \frac{1}{s_f} \int_0^{s_f} (|\kappa(s)|^2 + c_{obs}(s))ds \tag{16}$$

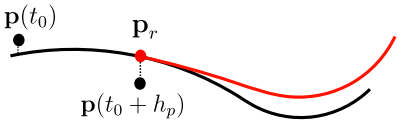



Fig. 5: Replanning scheme with $C^2$ continuity.

where $s_f$ is the total length of the path and the term $c_{obs}(s)$ avoids the optimized path to intersect with the obstacles.

Note that the curvature $\kappa(s)$ does not have an analytic form. Therefore the cost is computed using the forward Euler method. Parameters are then optimized through Subplex [17]. Subplex is a subspace-searching simplex method that requires no derivatives. It is well suited for noisy objective functions, thus is robust against errors caused by numerical integration of $|\kappa(s)|^2$. The time complexity of Subplex is linear to the number of parameters to be optimized. Another specificity of Subplex is that the cost is at least non-increasing through iterations. Hence we can run the optimization in a best-effort basis. We fix a constant maximal execution time for the algorithm. The algorithm returns the best result so far once the time is expired.

### *C. Initialization and Replanning*

Once the autonomous mode is activated, the current vehicle state $X(0) = (\left( \begin{smallmatrix} x_0 \\ y_0 \end{smallmatrix} \right), v_0, \phi_0, k_0)$ is used as the initialization parameter for the motion planner. The starting point for the path planner is set as $\mathbf{p}_0^* = \mathbf{w}_0 = \left( \begin{smallmatrix} x_0 \\ y_0 \end{smallmatrix} \right)$. The tangent of the starting point is set as the vehicle heading multiplied by a parameter $d_1$ representing the magnitude of the tangent: $\mathbf{t}_0 = d_1 \left( \begin{smallmatrix} \cos(\phi_0) \\ \sin(\phi_0) \end{smallmatrix} \right)$. In this paper we use a heuristic to select $d_1$ as half the length of $\mathbf{e}'_0$ ($d_1 = \frac{||\mathbf{e}'_0||}{2}$), the first piece of the piecewise linear path. An alternative is to consider $d_1$ as a free parameter in the optimization process mentioned in Section III-B. After deciding the starting point and starting tangent, we still need to fix the starting second derivative $\mathbf{a}_0$. In order to simplify mathematical derivation, we first rotate the coordinate system to transform $\mathbf{t}_0$ to $\tilde{\mathbf{t}}_0 = \left( \begin{smallmatrix} d_1 \\ 0 \end{smallmatrix} \right)$ such that $\tilde{\mathbf{t}}_0 = R(-\phi_0)\mathbf{t}_0$. Note that the trajectory is not affected as Bézier curve is invariant to affine transformation. Then from (9) we have $k_0 = \frac{\tilde{\mathbf{t}}_0 \times \tilde{\mathbf{a}}_0}{|\tilde{\mathbf{t}}_0|^3}$. We represent $\tilde{\mathbf{a}}_0$ in polar coordinate form such that $\tilde{\mathbf{a}}_0 = d_2 \left( \begin{smallmatrix} \cos(\phi_a) \\ \sin(\phi_a) \end{smallmatrix} \right)$,we then obtain

$$\begin{aligned} \sin(\phi_a) &= \frac{k_0 d_1^2}{d_2}, \\ \cos(\phi_a) &= \sqrt{1 - \sin^2(\phi_a)}. \end{aligned} \tag{17}$$

The unknown parameter $d_2$ represents the magnitude of the second derivative vector. It is heuristically set to be identical to $d_1$. The starting second derivative is then obtained by rotating back the coordinate system such that $\mathbf{a}_0 = R(\phi_0)\tilde{\mathbf{a}}_0$. As the vehicle moves forward, the perception system obtains new information that renders the current path obsolete. Hence the path needs to be replanned in a continuous way. In Fig. 5, the path of the last iteration is displayed in black while the new path is displayed in red. Let $\mathbf{p}(t_0 + h_p)$ be the current

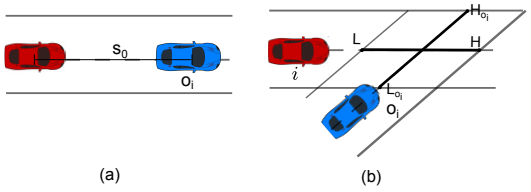


Fig. 6: (a) Obstacle following. (b)Intersection.

vehicle position such that $h_p$ is the replanning interval. Let $\mathbf{p}_r$ be the closest point to the vehicle on the reference path. If $||\mathbf{p}(t_0+h_p)-\mathbf{p}_r||$ is less than a threshold value $d_{thresh}$, we consider the vehicle is closely tracking the reference path. The replanning process then starts from $\mathbf{p}_r$, the tangent and second derivative vectors at this point are readily available by differentiating the path at $\mathbf{p}_r$. Such method ensures the $C^2$ continuity of the path during replanning. On the other hand, if $||\mathbf{p}(t_0+h_p)-\mathbf{p}_r||$ is larger than a threshold value $d_{thresh}$, we consider that the vehicle fails to track the reference path due to some reason, the path planner is then re-initialized using the updated vehicle state $X(t_0+h_p)$.

## IV. Velocity Planner

With the reference path as input, this section presents the generation of velocity profile using MPC.

### A. Safe Handling of Dynamic Obstacles

The dynamics of ego vehicle is described by (4). Let $t_m$ be the internal time of the MPC controller of the ego vehicle, a dynamic obstacle $o$ on the road is described as a point mass evolving along its path, maintaining constant velocity $v_o$ in the look-forward horizon of MPC. Therefore

$$s_o(t_m) = v_o t_m \tag{18}$$

where $s_o$ is the curvilinear abscissa of the obstacle along its path.

We consider two scenarios:

*1) Obstacle following:* If the perception system detects an obstacle and the behavior planner decides to stay behind the obstacle, we consider it as an obstacle following case: see Fig. 6 (a). We require the vehicle to remain brake-safe against the obstacle in front, that is: the ego vehicle should be able to maximally brake to avoid colliding on the obstacle, if the obstacle performs an emergency brake. The equation reads:

$$s_0 + s(t_m) + \frac{v^2(t_m)}{2|u_{min}|} < s_o(t_m) + \frac{v_o^2(t_m)}{2|a_{o,min}|} + s_{min} \tag{19}$$

such that $s_0$ is the initial distance, $a_{o,min}$ is the estimated maximal deceleration (minimal acceleration) of the obstacle and $s_{min}$ be the minimal safe distance.

*2) Intersection:* If the predicted path of the dynamic obstacles crosses the reference path of the ego vehicle, we consider it as an intersection scenario: see Fig. 6 (b). Let $[L,H]\times[L_o,H_o]$ be the collision region in the ego vehicle centered frame, such that only one vehicle (obstacle) is allowed to be in the region for any given time. We note that the ego vehicle has two choices, either to cross the region *before* the dynamic obstacle does, or *after* it. This actually implies the existence of homotopy classes of trajectories in on-road motion planning, as discussed in [18], [19]. We assume that the behavior planner decides the right-of-way of the ego vehicle. If the ego vehicle does not have the right-of-way, we enforce the brake-safe constraint:

$$s(t_m) + \frac{v^2(t_m)}{2|u_{min}|} - L < 0, \text{ if } s_o(t_m) < H_o, \tag{20}$$

On the other hand, if the ego vehicle has the right-of-way to cross first, then the ego vehicle should ensure the brake-safe property (see [18]) of the dynamic obstacle against the ego vehicle. The constraint read:

$$s(t_m) - H > 0 \text{ if } s(t_m) + \frac{v_o^2(t_m)}{2|a_{o,min}|} > L_o, \tag{21}$$

### B. MPC Formulation

Next we show how the velocity profile generation problem can be formulated as an MPC problem. Let $\xi = (s,v)'$ be the vehicle state. The vehicle model is discretized with an interval $h_v$ such that

$$\xi[k+1] = f^d(\xi[k], u[k]), \tag{22}$$

where the index $k$ represents MPC internal time $t_m = kh_v$.

Let $\boldsymbol{\xi} = [\xi[0],...,\xi[H]]$, $\mathbf{u} = [u[0],...,u[H]]$ respectively denote the vector of states and inputs, where $H$ is the prediction horizon. The MPC problem can then be formulated as a constrained optimization problem:

$$\min_{\mathbf{u}} \mathcal{J}(\boldsymbol{\xi}, \mathbf{u}), \tag{23}$$

subject to

$$\xi[0] = (s_0, v_0)', \tag{24}$$

$$\xi[k] \in \mathbb{R}^+ \times [0, v_{max}], u[k] \in [u_{min}, u_{max}], k = 0,...,H, \tag{25}$$

$$\xi[k+1] = f^d(\xi[k], u[k]), k = 0,...,H-1, \tag{26}$$

$$\forall o_i \in O, d(s[k], s_{o_i}[k]) < 0, k = 0,...,H, \tag{27}$$

$$v^2[k]\kappa[k] - a_{l,max} < 0, k = 0,...,H. \tag{28}$$

The cost function $\mathcal{J}(\boldsymbol{\xi}, \mathbf{u})$ varies with regards to current motion goal. A cost function for free driving is given as

$$\mathcal{J}(\boldsymbol{\xi}, \mathbf{u}) = \sum_{k=0}^{H} c_1(v_{target} - v[k])^2 + c_2 u[k]^2 \tag{29}$$

such that the vehicle tracks a target speed. Additional terms (fuel consumption, jerk, *etc.*) can also be integrated into the cost functional.

The cost function can also be configured to handle cooperative platoon driving. A constant time headway policy for platoon driving can be expressed through the following cost function

$$\mathcal{J}(\boldsymbol{\xi}, \mathbf{u}) = \sum_{k=0}^{H} c_1(s[k]+v[k]t_{des}-s_{prev}[k])^2+c_2 u[k]^2, \tag{30}$$

TABLE I: Parameters for path generation experiments

| | $\Delta s$ (m) | $\Delta l$ (m) | $N$ | $K$ | $w_1$ | $w_2$ |
|---|---|---|---|---|---|---|
| lane keeping | 10 | 0.15 | 10 | 41 | 0.5 | 1 |
| lane change | 10 | 0.3 | 10 | 41 | 0.5 | 1 |

where $s_{prev}[k]$ is the predicted position of the previous car at time step $k$ and $t_{des}$ is the desired time gap.

The safety constraints are written compactly in (27). In practice, they will be written as soft constraints to allow slight constraint violations in exchange of optimization feasibility. The safety condition should hold against all vehicles that have right-of-ways, *i.e.*, $\forall o_i \in O$. The constraints (28) limit the lateral acceleration of the ego vehicle.

The velocity profile can then be extracted from the optimal state vector $\boldsymbol{\xi}^*$. The reference trajectory is then fed to the low level controller for tracking.

## V. Experimental Results

The first set of tests focuses on the path planner. We test its capability of generating smooth trajectories for lane keeping and lane change scenarios. The second set of tests aims at verifying the on-road driving capability of the motion planner. Scenarios of lane keeping and intersection are examined. The last experiment tests the cooperative capability of the framework using platooning as an example. Execution time information is provided to verify the real-time ability. Experiments are conducted in a high-fidelity robotic simulator Webots [20] installed on a laptop with Core i5-4200M CPU and 8 GB RAM. The simulated vehicle model captures real vehicle dynamics such as steering dynamic response and friction of the tires. Algorithms are written in C++. For optimization algorithms, we use the Subplex and SLSQP solver available in the open-source *nlopt* package [21]. The video for the on-road driving experiments is provided as supplementary material.

### A. Path Generation

In Fig. 7a, we present a challenging lane keeping scenario in which three obstacles (red color) are distributed along the lane. Waypoints that representing the optimal piecewise linear path are labeled by blue "x", while paths before and after optimization are respectively labeled by red and green color. We observe that paths successfully avoid all obstacles. In Fig. 7b, we observe that the curvature after optimization is smoother. Same analysis can also be applied to the lane change scenarios in Fig. 7c and Fig. 7d. Note that different cost functions are used in two scenarios. Parameters for the experiments are available in Table I.

### B. On-road Driving capability

We demonstrate the capability of our framework in handling various driving scenarios. The path planning parameters of the following tests are identical to the first row of Table I. The sampling time for the velocity planner is selected as $h_v = 0.32s$.

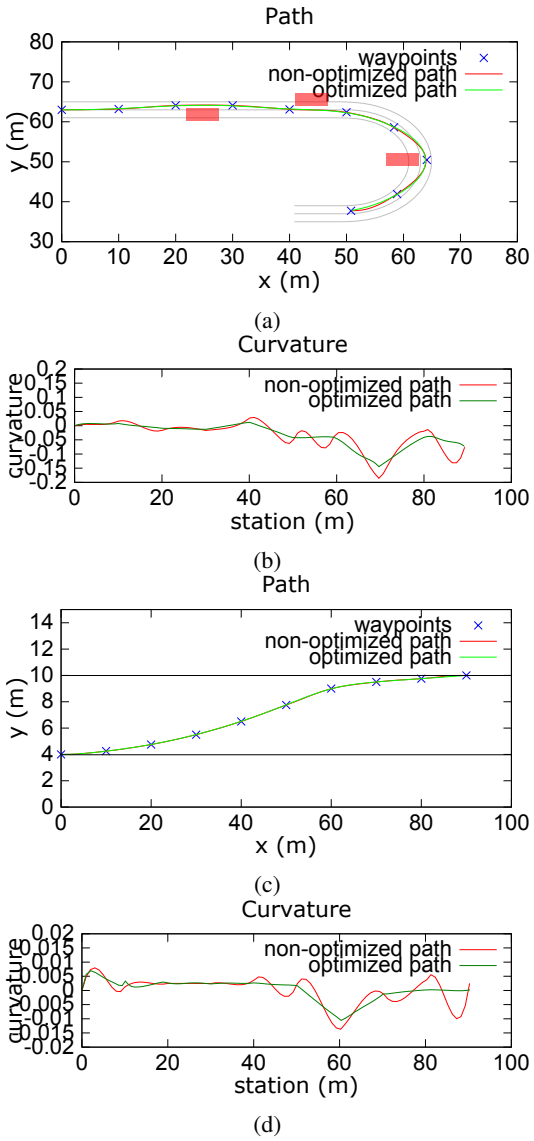
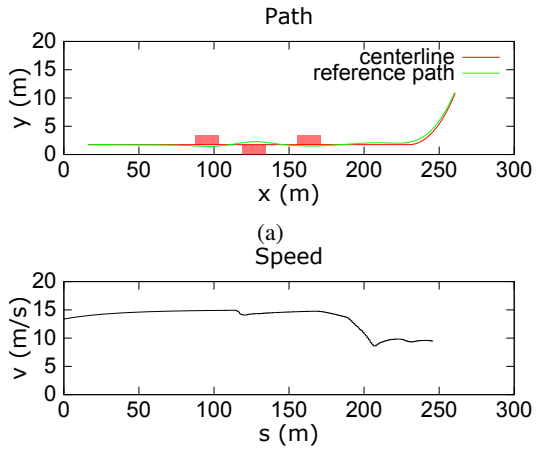



Fig. 7: Path planning for lane keeping and lane change scenarios

*1) Lane Driving with Static Obstacles (LD):* We first test the basic lane keeping capability of the proposed framework. The lane is composed by a straight segment and a curvy segment. Three obstacles are distributed along the lane (see Fig 8a). We adopt the cost function (29) for velocity planning with $c_1 = 1$ and $c_2 = 5$. From Fig 8a, we observe that the reference path is continuous and obstacle-free. The speed profile in Fig 8b shows that the vehicle slows down in advance before entering the curvy segment, thanks to the predictive ability of MPC.

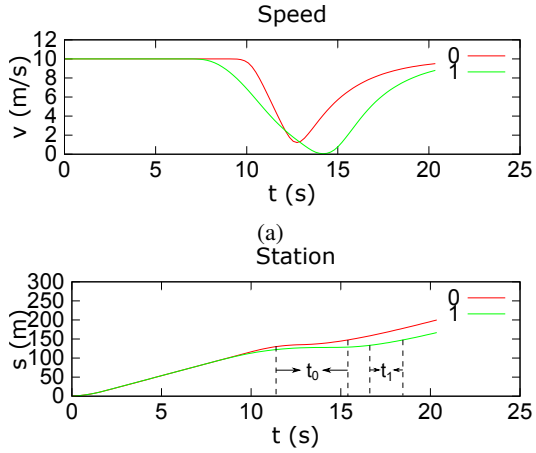
*2) Intersection Crossing with Emergency Brake of the Dynamic Obstacle (IC):* We consider the case that the ego vehicle confronts another vehicle with right-of-way at an intersection. The initial distances of two vehicles to the intersection are identical. We examine the response of the ego vehicle when the dynamic obstacle unexpectedly brakes. We adopt the cost function (29) for velocity planning with $c_1 = 1$, and $c_2 = 5$. From Fig. 9, we observe that the




(a)

(b)

Fig. 8: (a) Comparison of the centerline and the reference path. (b) Vehicle speed as a function of position




(a)

(b)

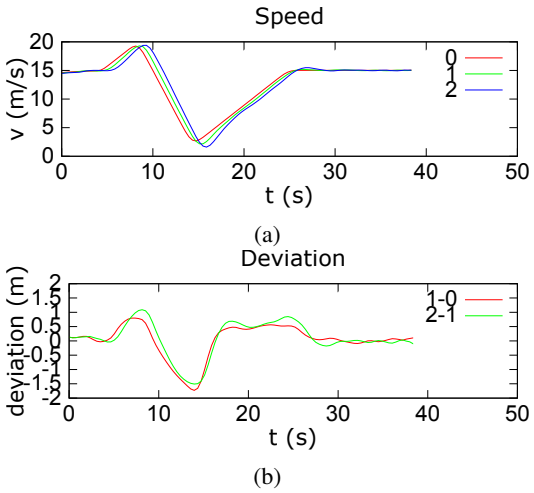
Fig. 9: Vehicle 0 has priority while vehicle 1 does not. Vehicle 0 performs an emergency brake from 11s to 13s. (a) Speed profiles of two vehicles. (b) Station of two vehicles. $t_0$ and $t_1$ are respectively two intervals that two vehicles are in the intersection. We note that two intervals do not overlap.

ego vehicle also brakes to avoid colliding on the dynamic obstacles and no collision occurs.

### C. Cooperative Platoon Driving (PC)

Vehicle platoon can be formed thanks to the flexibility of our framework. We simulate the platoon driving of three vehicles in a straight lane segment. The velocity profile of the leader vehicle is prefixed to have a perturbation between $9s$ and $28s$. We adopt the cost function from (30) for velocity planning with $c_1 = 1$ and $c_2 = 5$. The desired time gap is $0.5s$. Fig. 10a shows the speeds of three vehicles as a function of time. Fig. 10b shows the gap deviations from the desired gap. We observe that the gap deviation recovers to zero after the perturbation. The simulation confirms that our framework is able to handle platoon driving.

One highlight of the proposed framework is its efficiency. The dynamic programming part takes around $30ms$ and the




(a)

(b)

Fig. 10: The leader, the first follower and the second follower are labeled by natural numbers in ascending order. We introduce a perturbation to the leader between 9s and 28s to observe the responses of followers. (a) Speed profiles of three vehicles. (b) Deviations of inter-vehicle distances from the desired distances.

TABLE II: Execution time statistics for the velocity planner in different driving scenarios. LD–Lane Keeping Scenario. IC–Intersection Scenario. PC1–First follower in platooning scenario. PC2–Second follower in platooning scenario.

| | LD | IC | PC1 | PC2 |
|---|---|---|---|---|
| Average (ms) | 0.81 | 4.71 | 2.34 | 2.31 |
| Max (ms) | 4.13 | 7.68 | 9.61 | 10.99 |
| Min (ms) | 0.25 | 2.01 | 0.51 | 0.52 |
| Standard Deviation (ms) | 0.56 | 1.12 | 1.05 | 1.28 |

optimization algorithm for path planning is running in a best-effort basis with $25ms$ as the maximal execution time. The MPC problem for velocity planning is solved in millisecond scale. Table II summarizes the execution time statistics of the velocity planner for the different on-road driving test cases that we have performed on the above. The execution time is significantly less than the re-planning interval $h_p = 200ms$.

## VI. Conclusions

In this paper, we presented a novel motion planning framework for on-road autonomous driving in response to the requirements of flexibility, cooperative-awareness and efficiency. We adopted a two-step approach to split the motion planner into a path planner and a velocity planner. The path planner generates a curvature-continuous path using dynamic programming and quintic Bezier curve interpolation, while the velocity planner uses MPC to parameterize the path with an optimal velocity profile.

In order to better demonstrate the capabilities of our framework, we are planning to implement our framework on a real vehicles. We plan to use different platforms of the EU project AutoNet2030, which consist of automated cars, automated trucks and manually driven cars. We will also consider more complex traffic scenarios. Lastly, we want to benchmark our framework against other motion planning algorithms.

## REFERENCES


[1] A. De La Fortelle, X. Qian, S. Diemer, J. Grégoire, F. Moutarde, S. Bonnabel, A. Marjovi, A. Martinoli, I. Llatser, A. Festag, *et al.*, "Network of automated vehicles: the autonet2030 vision," in *21st World Congress on Intelligent Transport Systems*, 2014.

[2] M. McNaughton, C. Urmson, J. M. Dolan, and J. W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, pp. 4889–4895, 2011.

[3] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[4] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, "Efficient Sampling-Based Motion Planning for On-Road Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, 2015.

[5] B. Lau, C. Sprunk, and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2427–2433, 2009.

[6] D. González, J. Perez, R. Lattarulo, V. Milanés, and F. Nashashibi, "Continuous curvature planning with obstacle avoidance capabilities in urban scenarios," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pp. 1430–1435, IEEE, 2014.

[7] T. Gu and J. Dolan, "Toward human-like motion planning in urban environments," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pp. 350–355, June 2014.

[8] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha - A local, continuous method," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 450–457, 2014.

[9] J. Nilsson, Y. Gao, A. Carvalho, and F. Borrelli, "Manoeuvre generation and control for automated highway driving," in *The 19th world congress of the international federation of automatic control (IFAC)*, 2014.

[10] K. Kant and S. W. Zucker, "Toward Efficient Trajectory Planning: The Path-Velocity Decomposition," *International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.

[11] B. Nagy and A. Kelly, "TRAJECTORY GENERATION FOR CAR-LIKE ROBOTS USING CUBIC CURVATURE POLYNOMIALS Bryan Nagy and Alonzo Kelly," 2001.

[12] C. Chen, Y. He, C. Bu, J. Han, and X. Zhang, "Quartic bezier curve based trajectory generation for autonomous vehicles with curvature and velocity constraints," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 6108–6113, May 2014.

[13] C. Alia, T. Gilles, T. Reine, and C. Ali, "Local trajectory planning and tracking of autonomous vehicles, using clothoid tentacles method," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 674–679, IEEE, 2015.

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

[15] W. Xu, J. Pan, J. Wei, and J. M. Dolan, "Motion planning under uncertainty for on-road autonomous driving," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2507–2512, IEEE, 2014.

[16] 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 *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 1160–1166, IEEE, 2015.

[17] T. Rowan, "Functional stability analysis of numerical algorithms," p. 218, 1990.

[18] X. Qian, J. Gregoire, A. de La Fortelle, and F. Moutarde, "Decentralized model predictive control for smooth coordination of automated vehicles at intersection," in *Control Conference (ECC), 2015 European*, pp. 3452–3458, July 2015.

[19] P. Bender, O. S. Tas, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 1386–1392, IEEE, 2015.

[20] O. Michel, "Webots: Professional mobile robot simulation," *Int. Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[21] S. G. Johnson, "The nlopt nonlinear-optimization package,"