Generic Trajectory Planning Algorithm for Urban Autonomous Driving

Thibaud Duhautbout, Reine Talj, Véronique Cherfaoui, François Aioun, Franck Guillemard

Generic Trajectory Planning Algorithm for Urban Autonomous Driving

Thibaud Duhautbout[1,2], Reine Talj[1], Véronique Cherfaoui[1], François Aioun[2], Franck Guillemard[2]

***Abstract*— In this paper, a new local planning algorithm for urban autonomous driving is presented. Our main contribution is to define a fully algorithmic method, based on a geometrical representation of the environment, to compute predictive speed profiles on multiple paths, to ensure safety and comfort with respect to the scene and its predicted evolution. Simulation results are provided to evaluate the behaviour of the proposed algorithm on various scenarios. Those results show a good, comfortable and safe reaction of the vehicle to its static and dynamic environment with processing times compatible with real-time control.**

I. INTRODUCTION

Autonomous driving is a complex task and remains a challenge for industrial and academic actors. While more and more advanced assistance systems become available on highways, urban navigation is still highly manual. Urban environment is very complex: the network is dense, the roads can be narrow and have sharp turns, lots of driving rules have to be respected and the space is shared with other users, such as cars, pedestrians, bicycles, etc. To handle this complexity, the autonomous driving system has to be divided in different modules working together, each of which realizing a particular task. This paper describes a generic planning architecture, focuses on a local planning method which uses a path and velocity decomposition, and introduces an algorithmic speed planning method to improve existing path planning approaches.

In Section II, existing works about motion planning are covered. This work focuses entirely on a local trajectory planning algorithm, which has to be included in a larger architecture described in Section III. The proposed method is presented in Section IV and some results are discussed in Section V. Finally, a conclusion and some thoughts about future works are given in Section VI.

II. RELATED WORKS

Motion planning for self-driving cars has been widely covered in the literature, with various techniques and algorithms [1]. Methods based on random trees have been used for urban navigation [2], using incremental sampling to find a path from the initial position of the vehicle to a goal point. Each sampled configuration is checked for collision with the environment, so that only valid positions are added to the tree. Lattices planners [3] make use of a graph structure and efficient search algorithms to find a valid path to a goal point in a given environment. Both of these methods perform a direct search of one final path with static obstacle avoidance and do not account for moving obstacles. In [4], a temporal dimension is added to the lattice to allow the vehicle to adapt its speed to moving obstacles. However, the corresponding speed profile depends strongly on the time-discretization of the lattice.

Recent works try to solve the self-driving problem with an end-to-end machine learning process. In [5], a neural network is trained to steer the vehicle directly from a camera sensor. However, it is hard to prove that such methods always provide appropriate results. To provide interpretable results, [6] train a neural network to compute a cost volume from raw sensor data. A set of candidate trajectories is then sampled and evaluated with this cost volume to select the optimal trajectory. Although this provides interesting results, large amounts of labelled data are required to train the networks.

The tentacles method presented in [7] explicitly generates a fixed number of candidate paths and evaluates them against the environment. The best path is then tracked by the controllers for a short time period and the process is restarted. Initially represented by circle arcs, [8] enhances the method by constructing paths with clothoid arcs to ensure curvature continuity. The clothoid shape is well-suited to account for vehicle dynamics, but is not consistent with the behaviour of a vehicle on urban roads because most of the curves end up out of the road. In [9], [10] and [11], the paths are constructed as "parallel tentacles" and are defined by a lateral offset to the reference road, usually the center of the current lane. In [12], moving obstacles are considered by being extended in the underlying occupancy grid according to their heading and speed to provide some anticipation with respect to their expected motion. In these methods, the speed is not explicitly defined along the path.

In [13], speed values are defined on each point of the path depending on the curvature of the path, the speed of a preceding vehicle and the applicable speed limit. In [14], an hybrid trajectory planner for static environment is introduced, which separates the path and speed planning in two distinct tasks. In [15], the method is improved to consider moving obstacles in highway scenarios. Both of these methods use numerical optimization to compute speed profiles. [16] considers the urban environment by adapting the speed to the predictions of the moving obstacles. In this work, a similar approach is used: the "parallel tentacles" path-planning algorithm is used to sample candidate paths on the road, and a discrete adaptation method is introduced to compute a safe and comfortable speed profile on each

[1]Université de technologie de Compiègne, CNRS, Heudiasyc (Heuristics and Diagnosis of Complex Systems), CS 60 319 - 60 203 Compiègne Cedex (name.surname@hds.utc.fr)

[2]Stellantis, Centre Technique de Vélizy, Route de Gisy, 78140 Vélizy-Villacoublay (name.surname@stellantis.com)

This work was realized under cooperation contract between Stellantis and Heudiasyc laboratory.

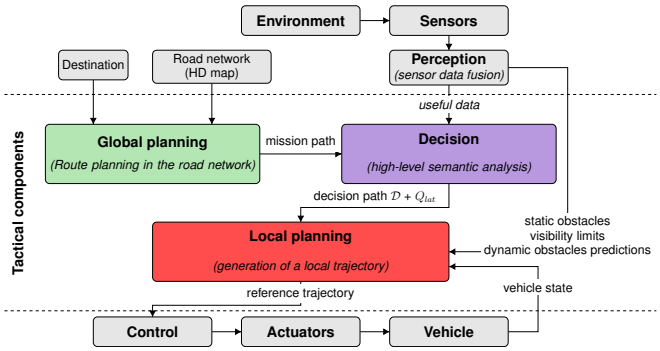

Fig. 1: Global architecture

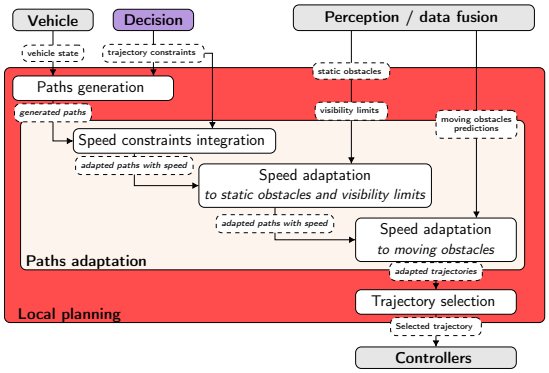

Fig. 2: Architecture of the proposed local planning method

path, by taking into account static and dynamic obstacles, and visibility limits. This algorithm is generic and adapted to different urban situations.

III. PLANNING ARCHITECTURE

As explained in the introduction, this work focuses on a local planning algorithm. However, it is clear that local planning alone can't provide all necessary functions for an autonomous vehicle. Figure 1 presents the complete architecture considered in this article, in which local planning is integrated. The dashed lines materialize the limits of the tactical components: global planning, decision and local planning. The aim of each of these components and the assumptions made are detailed in the following. Sensor data acquisition and processing, as well as vehicle control, are not covered in this paper. It is assumed that perception modules provide the data required for the tactical layer, and that efficient and predictive controllers are available to track the generated trajectories.

The *global planning* module is used to define the route from the starting point to the destination. The output of this module is called the **mission path** and contains the geometrical description of the route. If an HD map is available, the mission path can be enhanced with *a priori* information about the lane to use, the maximum allowed speed, expected signalization and any other useful information.

The *decision* module is the high-level reasoning module of this architecture. Its role is to define the **decision path**, representing the local path to follow to achieve the mission path. This path is an ordered set of $N^{\mathcal{D}}$ points defined by Equation 1. Each point P_d is defined in a global 2D reference frame with coordinates x and y, heading θ and curvature κ, and contains a speed value v^{dec} representing the maximum speed allowed, determined from the analysis of the situation (speed limit, stop signs, traffic lights, priorities...).

$$\mathcal{D} = \left\{ P_d = \left(x, y, \theta, \kappa, v^{dec}\right),\ d \in [1, N^{\mathcal{D}}] \right\} \quad (1)$$

The decision module also defines a lateral displacement interval $Q_{lat} = [q_{min}, q_{max}]$ which defines the drivable space around $\mathcal{D}$ and allows or forbids the vehicle to overtake.

Finally, the *local planning* module is responsible for the generation of a **reference trajectory** which will be tracked by the control system. This trajectory is a timely discretized representation of the path and speed profile to follow. The local planning tries to respect the lateral and speed constraints given by the decision, avoid static and moving obstacles, ensure dynamical feasibility and safety and maximize lateral and longitudinal comfort for the passengers.

The three planning modules do not function at the same rate. The global planning module can run at a very low frequency, or on-demand when the initial mission path can't be followed. The decision module should run at a moderate frequency to provide responsiveness to complex situations while keeping consistent and stable constraints for the local planning, which should run at a high frequency to adapt the reference trajectory to fast changes in the environment. In this paper, we detail the local planning module.

IV. PATH AND SPEED PLANNING ON TENTACLES

Method overview

The proposed planning algorithm is derived from the tentacles method. Figure 2 illustrates the different steps of the proposed method, each of which being described in the next sections. As in [14] and [15], path and speed planning are separated. A set of local candidate paths is generated from the trajectory constraints and the vehicle state. On each generated path, a speed profile is defined and adapted with respect to the environment to produce feasible and safe candidate trajectories. Defining a speed profile is key to provide anticipation and comfort for the passengers. Finally, the best trajectory is selected and provided to the controllers. All this process is executed at regular time steps to integrate new perception data.

A. Paths generation

As proposed in [11], candidate paths are composed of a transition part followed by a part parallel to $\mathcal{D}$, which seems close to a human behaviour.

The final points of the transition parts are defined by lateral offsets q_i from $\mathcal{D}$ and longitudinal offsets l_j from X_0, as in [17]. The lateral offsets are used to define multiple alternatives to avoid eventual obstacles, and are sampled in the Q_{lat} interval. The longitudinal offsets provide different transition shapes for each lateral offset, to generate more or less aggressive transitions. The l_j values can be fixed, or chosen relative to the vehicle speed. Figure 3 illustrates the

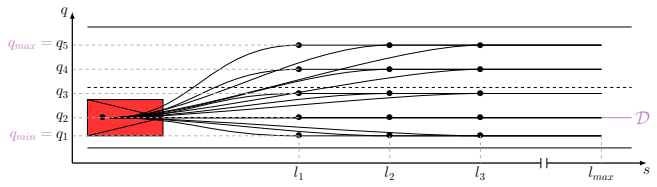

Fig. 3: Example of 15 generated paths around $\mathcal{D}$

result of the path generation process with 5 lateral and 3 longitudinal offsets.

The transition part is generated from the initial position of the vehicle to one final point as a parametric curve, with x and y being 5$^{\text{th}}$ degree polynomial functions. This representation is chosen because the paths can be computed by a direct linear system resolution. The parameters of each functions are determined to ensure position, heading, and curvature continuity at both extremities of the transition part.

Once the transition part of a path is generated, it is extended by a part parallel to $\mathcal{D}$ to reach a final length l_{max}. This value should be at least the same as the comfortable stop distance of the vehicle to provide enough anticipation. The transition and the parallel parts are discretized using a fixed number of points for each. Finally, a set of K paths $\{\mathcal{P}_k\}_{k\in[1,K]}$ is computed. Equation 2 defines the path $\mathcal{P}_k$ composed of $N_k^{\mathcal{P}}$ successive 2D points in a global reference frame. Each point holds speed v and acceleration a components which will be used in the adaptation steps.

$$\mathcal{P}_k = \left\{P_i = (x, y, \theta, \kappa, v, a, s)\,, i \in \left[1, N_k^{\mathcal{P}}\right]\right\} \qquad (2)$$

B. Paths adaptation

The paths adaptation steps (see Fig. 2) define a speed profile on each candidate path $\mathcal{P}_k$ to ensure safety and compliance with the given constraints. In these steps, the spatial components of $\mathcal{P}_k$ are fixed, and only the speed profile is adapted to satisfy the different constraints.

1) Speed constraints: The first step is to integrate speed constraints on the path. First, each point P_i of $\mathcal{P}_k$ is projected on $\mathcal{D}$ to retrieve the maximum speed v_i^{dec}. The speed is also adapted to the curvature of the path to maintain lateral comfort. Lateral acceleration a_{lat} is linked to v and κ by $a_{lat} = \kappa \cdot v^2$. Therefore, given a lateral acceleration bound $a_{lat}^{comf} > 0$, one can compute $v_i^{comf} = \sqrt{a_{lat}^{comf} \cdot |\kappa_i|^{-1}}$. Then, the speed value of P_i is set to $v_i = \min(v_i^{dec}, v_i^{comf})$.

2) Static obstacles and visibility limits: The second step is to define an eventual stop point linked to static obstacles and visibility limits. Figure 4 presents the algorithm used to adapt the speed on $\mathcal{P}_k$. Static obstacles, such as curbs or stopped cars, will not move and then require the ego-vehicle to fully stop to avoid a collision on the fixed path. For each point of the path, the algorithm checks for overlaps between the polygon of the ego-vehicle and the polygon representing the obstacles, given by the perception. The advantages of this geometric method are to be generic over the shape and the position of the obstacles, and to consider the full shape of the car. The *contact point* C is the last position

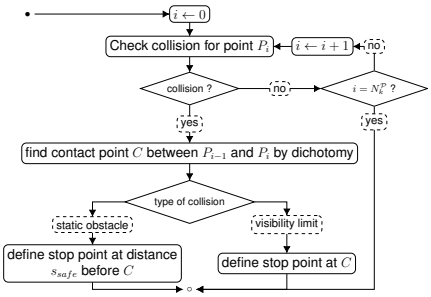

Fig. 4: Static obstacles and visibility adaptation process for $\mathcal{P}_k$. • is the entry point of the algorithm, and ○ the endpoint.

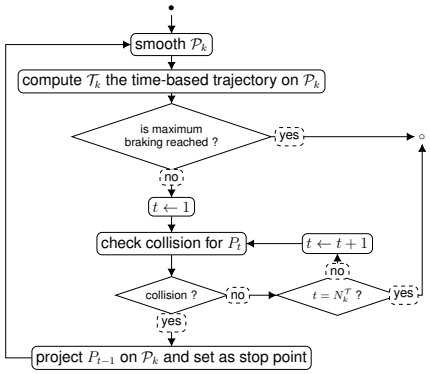

Fig. 5: Moving obstacles adaptation process for $\mathcal{P}_k$

without collision on $\mathcal{P}_k$. If the ego-vehicle stops there before an obstacle, it will not likely be able to maneuver due to non-holonomic constraints. To prevent this, a longitudinal safety distance s_{safe} is introduced to move back the stop point. Visibility limits are treated as static obstacles, but without safety distance to ensure security while limiting conservatism when driving in occluded turns. The stop point is then inserted on $\mathcal{P}_k$ with $v_i = 0$.

3) Moving obstacles: The third step of the adaptation concerns dynamic obstacles. The process for a path $\mathcal{P}_k$ is described on Figure 5. To predict the trajectory of the ego-vehicle, the speed profile on $\mathcal{P}_k$ needs to be smoothed, because no dynamic consideration about acceleration has been introduced yet. The smoothing algorithm is presented in the next section. From the smoothed speed profile on $\mathcal{P}_k$, a time-based trajectory $\mathcal{T}_k$ with a sample time δ_t can be computed, as shown on Equation 3.

$$\mathcal{T}_k = \left\{P_t = (x, y, \theta, \kappa, v, a)\,, t \in \left[1, N_k^{\mathcal{T}}\right]\right\} \qquad (3)$$

The value of δ_t must match the sample time of the predictions of moving obstacles. Here, collisions are checked against the predicted state of moving obstacles given by the perception module. If a collision is detected at P_t, the point P_{t-1} (which is collision-free) is set as stop point. Then, a new verification is required because the modification of the speed profile induces a new behaviour of the vehicle, which may collide with other moving obstacles. Figure 6 illustrates this iterative process with an overtaking example.

To ensure a safety distance between the ego-vehicle and moving obstacles, the shape of each actor used for the inter-

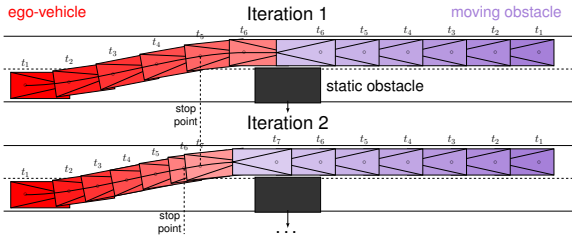

Fig. 6: Illustration of the adaptation of the speed profile with a moving obstacle. The ego-vehicle plans to overtake a static obstacle and a vehicle comes on the opposite lane. At iteration 1, a collision is detected at time t_6, the point at t_5 is then marked as stop point. At iteration 2, the vehicle slows down to stop at the right position but is still in collision with the moving obstacle at time t_7.

section tests is extended to a time period dt corresponding to the desired inter-vehicular time. To determine that there is no collision at time t, there must be no collision on the predicted evolution from time t to $t+dt$. This brings some anticipation in the planning process and induces safety distances when crossing or following other vehicles. Contrary to the second step of adaptation (IV-B.2), no dichotomy is performed here to precisely estimate the stop point, in order to save processing time.

This speed adaptation process can be seen as conservative. Indeed, the global policy induced is to stop and pass after all other vehicles. This policy is chosen to ensure safety and to plan solutions in the worst case. However, the reactive scheme allows to adjust the plan according to the evolution of the environment.

4) Speed smoothing: On a path, the speed profile is represented as a spatial profile, because the constraints we conider are linked to a certain position. Each point is defined by its curvilinear abscissa s_i and holds speed v_i and longitudinal acceleration a_i values. It is assumed that values of v_0 and a_0 are set from the current state of the vehicle. The following smoothing algorithm is used to create a speed profile which integrates a target longitudinal behaviour while ensuring the respect of the speed constraints set during the adaptation steps. It is based on three temporal profiles: comfortable acceleration, comfortable deceleration and maximal deceleration. Each profile is defined by a fixed acceleration and a variation rate. These profiles are illustrated on Figure 7.

The comfort acceleration profile is used when the car needs to accelerate. When the car needs to slow down without stopping, the comfort deceleration profile is used, but if the car needs to stop, the algorithm is allowed to find an optimal deceleration profile between the comfort and the maximal ones to ensure that the stop will occur at the right position. This optimization trades comfort for safety: in this case, in order to respect the stop point and avoid a potential collision, comfort requirements may be discarded.

The speed profile is smoothed in a two-pass iterative way. The first pass smoothes the accelerations: for each point (s_i, v_i, a_i) of the path, the maximum reachable acceleration

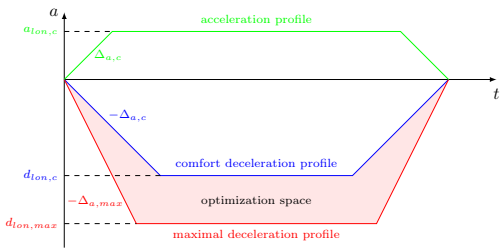

Fig. 7: Illustration of the different acceleration/deceleration profiles used

a_{i+1}^{max} and speed v_{i+1}^{max} at the next position s_{i+1} are computed from the appropriate profile. The next speed is then updated by $v_{i+1} \leftarrow \min(v_{i+1}, v_{i+1}^{max})$, and a_{i+1} is updated the same way. The second pass smoothes the decelerations, which are treated the same way as accelerations when the path is processed backwards. This algorithm makes it possible to define smooth speed and acceleration profiles and to anticipate the decelerations before a stop or a sharp curve.

The smoothing algorithm also considers the initial speed of the vehicle to ensure the generation of a coherent profile. Due to that, in some cases, speed constraints may be impossible to meet, and the resulting profile may present higher speeds than requested. This usually happens when processing a path with a sharp lateral transition, which creates high lateral acceleration values. To avoid consequent dangerous maneuvers, the selection step performs a check to filter these cases. Also, if a stop is requested too close from the initial position of the vehicle, the maximal braking speed profile will be computed but the stop constraint might not be met.

C. Trajectory selection

When all the candidate trajectories have been adapted, the algorithm needs to decide which one will be sent to the control systems. The first step is to remove the trajectories that could threaten the stability of the vehicle. The lateral acceleration is computed on each candidate trajectory, and if it exceeds a security bound a_{lat}^{sec}, the trajectory is discarded. In the case where all trajectories are discarded, the filter is considered irrelevant and all trajectories are kept for the rest of the selection.

The trajectories are then divided into 4 groups: 1) trajectories without collision; 2) trajectories without collision but which stop too close to static obstacles; 3) trajectories colliding with a static obstacle; 4) trajectories colliding with a moving obstacle. Only the trajectories of the first non-empty group are kept and all others are discarded. This gives a first layer of selection based on a safety principle.

Then, a cost function $\mathcal{C}$ is defined as:

$$\mathcal{C}(\mathcal{T}_k) = C_k + \delta_{k,1} \cdot \sigma_1 + \delta_{k,2} \cdot \sigma_2 \tag{4}$$

where the cost C_k is given by:

$$C_k = \delta_t \times \sum_{t=1}^{N_k^{\mathcal{T}}} \left({a_{lon,t}}^2 + {a_{lat,t}}^2 \right)$$

and the two decision variables $\delta_{k,1}$ and $\delta_{k,2}$ are defined by:

$\delta_{k,1} = 1$ if $\mathcal{T}_k$ leads to a stop, 0 otherwise

$\delta_{k,2} = 1$ if $\exists t \in [1, N_k^{\mathcal{T}}], |a_{lat,t}| > a_{lat}^{comf}$, 0 otherwise

In Equation 4, C_k represents the integral of the squared total acceleration of the trajectory $\mathcal{T}_k$, with $a_{lon,t}$ and $a_{lat,t}$ representing the longitudinal and lateral acceleration. σ_1 and σ_2 are high costs defined such that $\forall k, \sigma_2 > \sigma_1 > C_k$. Those costs are linked to the decision variables $\delta_{k,1}$ and $\delta_{k,2}$. $\delta_{k,1}$ activates the cost σ_1 to penalize stopping trajectories, and $\delta_{k,2}$ activates the cost σ_2 to penalize non-comfortable trajectories. Finally, the trajectory with the lowest cost value is selected and provided to the control modules.

D. Processing time optimization

All candidate paths are not systematically generated. To improve processing time, especially in easy situations, a first set of paths without lateral offset (i.e., with $q_i = 0$) is generated. Those paths are then processed by the adaptation steps previously described. If at least one resulting path does not lead to a full stop, the algorithm directly goes to the selection step. If all paths lead to a full stop, it means that the decision path can't be followed and that a lateral displacement might be required. A second set of paths with lateral offsets around $\mathcal{D}$ is then generated, adapted, merged with the first one and provided to the selection function.

V. RESULTS

Table I summarizes the theoretical differences between the proposed algorithm and the closest works of Section II.

Simulations have been conducted to study the results of the presented algorithm in different situations. The algorithm is implemented on MATLAB-Simulink® with code generation and tested on a laptop with an Intel® i7 processor running at 2.70GHz. The environment is manually defined, and is perfectly known by the vehicle. The global trajectory of the moving obstacles is also fully defined, and the exact future motion is used for the predictions. No sensors were simulated and no uncertainties were introduced in the scenarios.

The local planning algorithm is executed every 200ms and the reference trajectory is perfectly followed. For path generation, l_j values are taken in $\{5, 10, 15, 20, 25, 30\}$, and $l_{max} = 80m$ to cover the visibility range of the vehicle. 11 q_i values with a 0.5m lateral interval are used to cover the road, leading to a total of 66 paths. The paths are discretized with $N_k^{\mathcal{P}} = 100$ points, and the trajectories with $N_k^{\mathcal{T}} = 51$ points with $\delta_t = 0.1s$. Speed and acceleration constraints used are described in Table II.

Results are presented on 3 different environments: 1) a classic crossing with occulting obstacles, 2) a two-lanes roundabout and 3) a straight road with a static obstacle on the ego-vehicle's lane. In the following figures, better viewed on the PDF version of the paper, the blue lines represent the center of the different lanes of the network. A simple implementation of the decision uses a local portion of the mission path given by the scenario to compute $\mathcal{D}$. The gray areas represent the static obstacles and the outside

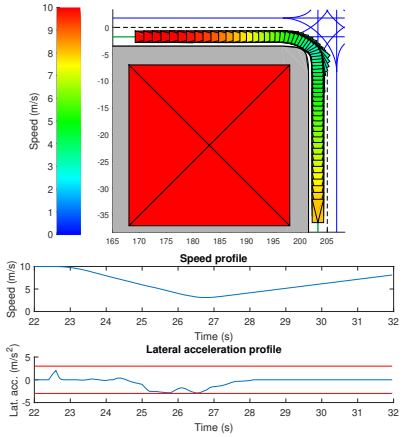

Fig. 8: Right turn with low visibility

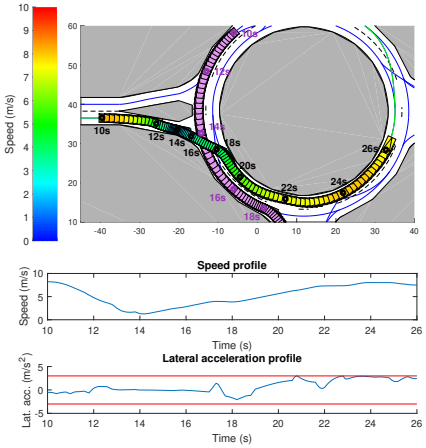

Fig. 9: Roundabout merging with one moving obstacle (purple)

of the road, and the red squares represent obstacles limiting the visibility of the vehicle. The trajectory followed by the vehicle is displayed with a color ramp indicating the speed.

The first environment presents the behaviour of the vehicle without moving obstacles and with occlusions in the turns. As can be seen on Figure 8, the speed of the vehicle decreases before entering the turn due to the limited visibility and to respect the lateral acceleration bounds.

The second environment shows the results of the algorithm with curved roads with another vehicle (displayed in light purple). Figure 9 shows that the ego-vehicle slows down before entering the roundabout to let the other vehicle pass before. The dots on the trajectory show the position of each actor at the indicated time. When the way is clear, the vehicle reaches the maximum speed which respects the comfort bound.

The last environment presents a complex case where the ego-vehicle's lane is blocked by an obstacle, and two other vehicles come on the opposite lane. On Figure 10, the ego-vehicle starts an overtaking trajectory but then detects the other vehicles and comes back in its lane to stop before the static obstacle, and waits until the left lane is free to perform the overtaking.

Method	[15] (highway)	[16] (urban)	Proposed method
Path planning	Sampling-based with 5^{th} degree polynomial offset function	Sampling-based with 5^{th} degree Bezier curves	Sampling-based with 5^{th} degree spatial polynomial curves
Static obstacles adaptation	Not considered	Path validity checked by polygon overlap Trajectory shortened to stop before collision	Positions validity checked by polygon overlap Stop point defined to stop before collision
Moving obstacles adaptation	Generation of a safety corridor in the path-time space	Trajectory validity checked by spatial projection overlap Speed adjustment based on motion direction	Predicted position validity check by polygon overlap Stop point defined to stop before collision
Speed planning	Speed profile optimization Computed on all candidate paths	Speed smoothing based on speed constraints Computed only on the selected path	Speed smoothing based on speed constraints Computed on all candidate paths
Selection	Cost function over candidate trajectories	Cost function over candidate paths	Cost function over candidate trajectories

TABLE I: Comparison summary of the method

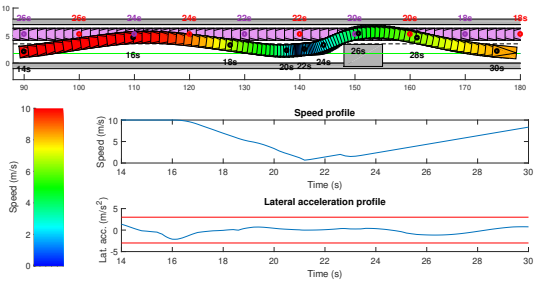

Fig. 10: Overtaking with moving obstacles on the left lane

Table III summarizes the processing time of the local planning step for the presented results. The scenarios with the highest processing times correspond to scenarios with moving obstacles, which are complex cases and thus take more time to process. The highest processing time obtained for the presented results is lower than the replanning time (200ms). Generally, the median processing time seems efficient enough to use the algorithm in embedded conditions.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, a new trajectory planning algorithm is presented. Based on the tentacle algorithm, it introduces explicit speed planning in urban environment by finding stop points with respect to static and dynamic obstacles, and by computing smoothed speed profiles on each candidate path. Simulations are carried out to analyze the behaviour of the algorithm and show encouraging results.

Future works will focus on studying the behaviour of the vehicle in a realistic closed-loop simulation, in challenging situations with more moving obstacles. Uncertainties, occlusions, and tracking errors will be introduced to study the strengths and limits of this algorithm.

longitudinal profile	**comfort**	**maximal**
target speed	$10m \cdot s^{-1}$	NC
acceleration	$1m \cdot s^{-2}$	NC
deceleration	$-2m \cdot s^{-2}$	$-10m \cdot s^{-2}$
variation	$3m \cdot s^{-3}$	$10m \cdot s^{-3}$
lateral bounds	**comfort**	**security**
lateral acceleration	$3m \cdot s^{-2}$	$5m \cdot s^{-2}$

TABLE II: Speed and acceleration arameters constraints used

Scenario	min time (ms)	median time (ms)	max time (ms)
1 (fig. 8)	2.6	3.6	6.5
2 (fig. 9)	3.9	7.2	51.7
3 (fig. 10)	3.1	4.9	76.6

TABLE III: Processing time distribution for the presented simulations

REFERENCES

[1] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, Nov. 2015.

[2] Y. Kuwata *et al.*, "Motion planning for urban driving using RRT," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nice: IEEE, Sep. 2008, pp. 1681–1686.

[3] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2005, pp. 1–7.

[4] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1879–1884.

[5] M. Bojarski *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[6] W. Zeng *et al.*, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[7] F. von Hundelshausen *et al.*, "Driving with tentacles: Integral structures for sensing and motion," *Journal of Field Robotics*, vol. 25, no. 9, pp. 640–673, Sep. 2008.

[8] M. Himmelsbach *et al.*, "Autonomous Off-Road Navigation for MuCAR-3: Improving the Tentacles Approach: Integral Structures for Sensing and Motion," *KI - Künstliche Intelligenz*, vol. 25, no. 2, pp. 145–149, 2011.

[9] S. Thrun *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, Sep. 2006.

[10] C. Urmson *et al.*, "Autonomous Driving in Urban Environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, Aug. 2008.

[11] K. Chu, M. Lee, and M. Sunwoo, "Local Path Planning for Off-Road Autonomous Driving With Avoidance of Static Obstacles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1599–1616, Dec. 2012.

[12] H. Mouhagir *et al.*, "Evidential-based approach for trajectory planning with tentacles, for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3485–3496, 2020.

[13] J. Kim, K. Jo, W. Lim, and M. Sunwoo, "A probabilistic optimization approach for motion planning of autonomous vehicles," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 232, no. 5, pp. 632–650, Apr. 2018.

[14] Y. Zhang *et al.*, "Hybrid trajectory planning for autonomous driving in highly constrained environments," *IEEE Access*, vol. 6, pp. 32 800–32 819, 2018.

[15] W. Lim, S. Lee, M. Sunwoo, and K. Jo, "Hybrid trajectory planning for autonomous driving in on-road dynamic scenarios," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2019.

[16] A. Artuñedo, J. Villagra, and J. Godoy, "Real-time motion planning approach for automated driving in urban environments," *IEEE Access*, vol. 7, pp. 180 039–180 053, 2019.

[17] X. Li, Z. Sun, Q. Zhu, and D. Liu, "A unified approach to local trajectory planning and control for autonomous driving along a reference path," in *2014 IEEE International Conference on Mechatronics and Automation*. Tianjin, China: IEEE, Aug. 2014, pp. 1716–1721.