Real-time planning for adjacent consecutive intersections

Fernando Garrido, David Gonzalez Bautista, Vicente Milanés, Joshué Pérez, Fawzi Nashashibi

Real-time planning for adjacent consecutive intersections

Fernando Garrido[1,2], David González[1], Vicente Milanés[1], Joshué Pérez[3] and Fawzi Nashashibi[1]

Abstract— **Real-time path planning constitutes one of the hot topics when developing automated driving. Different path planning techniques have been studied in both the robotics and automated vehicles fields trying to improve the trajectory generation and its tracking. In this paper, a novel local path planning algorithm combining both off-line and real-time generation for automated vehicles in urban environments is presented. It takes advantage of an off-line planning strategy to achieve a smooth and continuous trajectory generation with adjacent consecutive intersections. Therefore, this approach allows a smooth and continuous trajectory by means of planning consecutive curves concurrently, i.e., it will plan the upcoming curve and the following one in advance, taking into account the constraints of the infrastructure and the physical limitations of the vehicle. The real-time planning algorithm has been tested in simulation based on the INRIA-RITS vehicles architecture. The results obtained show an improvement in the smoothness, continuity and comfort of the generated paths.**

I. INTRODUCTION

Vehicle automation is gaining more and more attraction within the intelligent transportation research field. As a matter of fact, significant demonstrations of automated capabilities have been shown in the last years. Specifically, one of the most relevance was the Bertha Benz memorial route carried out by the Karlsruher Institut für Technologie (KIT) [1]. They emulated the first trip did by a Mercedes car but on a fully automated way on German roads, presenting some advances on perception, localization and motion planning. In 2010, the VisLab Intercontinental Autonomous Challenge was conceived as an extreme test of automated vehicles [2], where two driverless vehicles made a 16,000 kilometers trip from Parma (Italy) to Shanghai (China). Some important demonstrations have been celebrated last year, such as the ones related to the CityMobil-2 project and the 22nd ITS World Congress in Bordeaux, where automated vehicles were tested in urban areas in order to enhance the vehicle-infrastructure communication and that way the cooperation among them, pursuing the objective of building an efficient public transport system in dedicated areas [3]. Recently, several teams participated in the 2016 Grand Cooperative Driving Challenge (GCDC)[1]. It was focused on cooperative driving, where some important advances were made in sensor fusion, vehicle-to-vehicle communication and cooperative control. Finally, some promising results have been achieved in urban roads, handling a variety of dynamic scenarios while respecting the traffic rules [4]. Although these are very promising results, there is still a long way to go before having fully automated vehicles driving on public roads.

In the automated vehicles control field, the planning of the route plays a key role in order to improve transport efficiency and safety [5], increasing the driving stability [6]. For open spaces, such as the urban areas where automated vehicles deal with the different road actors (pedestrians, cyclists, etc), it has to determine the best feasible and collision-free path. Based on the team's vehicle architecture described in [7], path planning composes the decision stage of the architecture. This is divided in two sub-stages: global and local planning. The first one uses the information of the environment to create a path of way-points that describe a route itinerary. Meanwhile, the local planner ensures a real time, safe and comfortable trajectory throughout the path, smoothing the raw path by considering the kinematics of the vehicle and the road constraints.

Current planning systems consider a short term path planning, that is, with a short planning horizon. As examples, an horizon of 10 seconds was considered by the KIT institute in the Bertha Benz Memorial Path [1], whereas an horizon of 30 meters was considered by the Vislab team of Parma University in the Public Road Urban Driverless car test (PROUD) of 2013 [8] and the same distance was considered in the Boss car of Carnegie Mellon University (CMU) during the 2007 Darpa Urban Challege [9].

Digital maps and global planners are getting more and more precise, being able to provide information beforehand about the environment where the vehicle has to drive. That is, the way-points that define the intersections on the route are known in advance (both the distances to reach the intersections, their angles and their turning directions). That way, the optimal trajectory for each isolated curve [10] is already registered in databases. Using this previous knowledge, a path planning algorithm that optimizes the two following curves generating a smoother and continuous path is presented.

The rest of the paper is structured as follows. Section II presents a path planning techniques overview. The path planning algorithm for interpolated curves is explained in section III. This is the basis of the optimized path planning algorithm for consecutive curves, presented in section IV. Simulation results are included in Section V. Finally, some concluding remarks and future work are given in Section VI.

This work was supported by the VeDeCoM institute

[1]Inria Paris, Robotics and Intelligent Transportation Systems (RITS) Team, 2 Rue Simone IFF, 75012 Paris, France {fernando.garrido-carpio, vicente.milanes, david.gonzalez-bautista, fawzi.nashashibi} @inria.fr

[2]VeDeCoM institute, 77 Rue des Chantiers, 78000 Versailles, France fernando.garrido-carpio@vedecom.fr

[3]Tecnalia, Calle Geldo, Edificio 700, Parque Cientfico y Tecnolgico de Bizkaia, E-48160 Derio, Spain joshue.perez@tecnalia.com

[1]http://www.gcdc.net

II. Path planning techniques overview

There exist several path generation techniques that can be used in the planning stage. The most common ones in the automated navigation field can be classified in the following categories [11]:

Graph search based planners: Describe the environment as a grid or lattice of different states. They provide the states the vehicle would visit to achieve the destination. Then, these algorithms give a solution in the case of having a sequence of valid states, but this could not be the optimal one. Some examples are: 1) the A* algorithm that is described as an extension of the Dijkstra's algorithm [12]. It was used in the 2007 DARPA Urban Challange [13]; 2) The State Lattice that searches the discretization of the vehicle state space represented as a direct graph to find a motion plan that satisfies the constraints [14]. This method has been used in the generation of trajectories to be followed at high-speed running with real time performance [15], or to compute highly optimized paths between destinations [16].

Sampling based planners randomly sample the space looking for connectivity and fulfilling with timing constraints. As an example, the Rapidly-exploring Random Tree (RRT) method, which builds an incremental tree for searching in high-dimensional spaces. There exist an enhancement of this algorithm, called Enhanced RRT or RRT*. It quickly finds a feasible motion plan, but unlike RRT, RRT* almost surely converges to an optimal solution thanks to the asymptotic optimality property [17].

Interpolating Curve Planners: These methods generate a smooth path from a set of way-points that describe the road. Below the most significant methods are described.

Clothoids: These are curves whose curvature changes linearly with its curve length. Since they are defined in terms of Fresnel integrals that cannot be solved analytically, using them in real-time applications is a difficult issue in comparison with non-linear curvature curves that are easier to compute [18]. Clothoids are used in road design because they provide a constant change on the curvature, allowing a smooth change from linear to circular stretches and vice-versa.

Polynomial curves: These are curves defined by polynomial equations. They are used to fit a series of data points, possibly subject to constraints, where either interpolation or smoothing is involved.

Splines: These are piecewise differentiable curves defined by polynomials. These can be parametric equations or B-splines, i.e., a generalization of Bézier curves. Their knots or polynomial joints pose a high degree of smoothness.

Bézier curves: These are polynomial curves that use information of checkpoints, called control points, to be generated. The basis of these curves are the Bernstein poynomials.

A. Bézier curves: definition and properties

Among the different path planning techniques, we use Bézier curves because of their simplicity and low computational load which make it more suitable than the other path generation techniques [19], allowing a fast trajectory computation. This is a needed feature that has to be fulfilled by the proposed local planner in order to generate the selected trajectory from the database and send it to the controller stage. Thanks to be defined by a set of control points, the curves generated are malleable. That way, it is easier to join curves preserving the curvature continuity.

The generation of a n-degree Bézier curve is mathematically defined by Equation 1.

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i \quad t \in [0,1] \tag{1}$$

where n is the degree of the polynomial equation and P_i are the control points that define the curve.

In [19] some features of the Bézier curves are described. The most relevant for the work done are explained below.

- Bézier curves always start on the first control point P_0 and they end at the last control point P_n.
- The direction vector that defines the beginning of the curve is described by $\overrightarrow{P_0P_1}$ and the direction vector that defines the end of the curve is described by $\overrightarrow{P_{n-1}P_n}$.
- The curves are fully symmetric, i.e., if the control points are reversed, we get the same curve.
- The curves will be always framed within the convex hull defined by the control points, i.e., the one formed by the outermost control points.
- The behaviour of the curve concavity is consistent with the concavity formed by the control points.

Thus, we will test our trajectory generation approach in urban scenarios where different kind of curves can be found. The results obtained will be compared with the previous works developed in the framework of INRIA RITS team for intersection scenarios [20].

III. Vehicle path planning for interpolated curves

Urban areas usually present several consecutive intersections in a short period of time, with a limited space between them. Our approach proposes a strategy to generate a smoother real-time path planning. Considering the information of the route (coming from the global map), the road layout and the physical limitations of the vehicle, the optimal curve for each intersection can be load to generate a continuous and smoother path optimizing consecutive curves.

The proposed planning strategy tries to emulate the human behaviour when driving. Thus, when a human driver is driving on a well-known environment, he will take advantage of the road to adjust his trajectory depending on the road layout. Similary for the local planner, when several zig-zag curves are presented, the system will keep the vehicle in the internal side of the road, that is, not opening its trajectory followed, like a human driver does. By contrast, when several consecutive curves with the same direction of rotation are presented, it will open the trajectory followed by the vehicle in the turn, without returning to the centre of the road. This provides to the planning system a naturalistic driving style.

To achieve a real-time smooth and continuous path generation, the proposed algorithm is based on an off-line planning

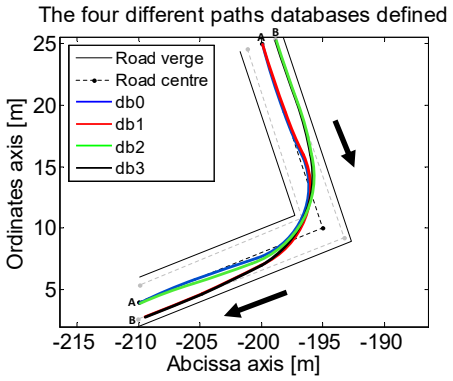

Fig. 1: Paths databases for consecutive intersections scenarios

TABLE I: Paths databases according to the different directions of rotation (d.o.r)

Trajectory	Starting point	Ending point
Blue	A	A
Red	A	B
Green	B	A
Black	B	B

strategy presented in [10]. Since the real-time computational cost for the path planning can be too high, an off-line evaluation has been applied in this module. It consists on the generation of a database of the best curves that a vehicle can follow on an interpolated intersection scenario, considering a big set of possible intersection scenarios while changing the angle of the curve and the length of the segments. The curve for each kind of intersection is chosen by means of the cost function Q described in Equation 2, which tries to minimize both the curvature (k) of the path and its derivative ($\dot{k}$), trying to remove abrupt changes on it.

$$Q = \sum_{i=0}^{n} w_{ki} \left|k_i\right| + w_{\dot{k}i} \left|\dot{k}_i\right| \tag{2}$$

As the maximum speed of the experimental vehicles (cybercars) is 40 kph [21], at this low-speed the vehicle behaviour is not significantly affected by vehicle dynamics and the planning is mainly affected by vehicle's kinematics and physical road constraints [21]. To that end, as every vehicle has a maximum steering angle, all the curves evaluated have to present a curvature less than or equal to the maximum curvature feasible by the vehicle. Furthermore, the curve is generated into the road limits by placing the control points into the polygon formed by an internal separation of the vehicle width with respect to the road limits, according to the 4th Bézier property, as explained in [10].

Table I explains the combination of all possible scenarios that can be found. The starting/ending points for generating the trajectory can be either center in the lane (represented by A) or slightly moved to the side of the lane (represented by B) according to a more naturalistic driving.

Fig. 1 summarizes the possible consecutive intersection scenarios that can be found in an urban scenario in terms of

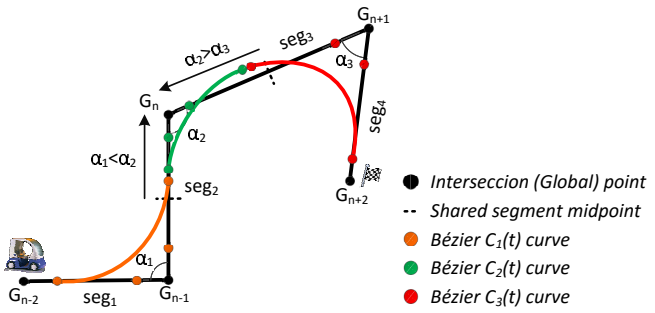

Fig. 2: Adjacent consecutive intersections planning strategy

their direction of rotation. For each scenario, a database is generated. They will be read by the local planner in order to optimize in real-time the trajectory generated by the consecutive curves. The trajectory marked in blue (A-A) represents a curve that connects to center-center lane trajectory because both curves have the same direction of rotation. In contrast, the red one (A-B) shows the connection center-side, where the trajectory finishes at the side of the lane because the next curve has the same direction of rotation than the last one. On the other hand, the green and black trajectories represent the scenarios side-center (B-A) and side-side (B-B), where the previous curve has the same direction of rotation than the upcoming one, so these trajectories start at the border of the lane. The green trajectory finishes at the middle of the lane because the next curve has different direction of rotation, whereas the black one finishes at one side of the lane because the next curve has the same direction of rotation.

IV. Optimized planning for consecutive curves

The main contribution of this work is the generation of a safe and smooth path planning with an extended planning horizon of two curves for urban scenarios, assuming a reliable information of the route coming from the global planner. A real-time planning is generated in order to decide which path is the optimal one. As it is computationally expensive analyzing all the possible paths that a vehicle could track, the local planner loads from the off-line generated database the isolated curves generated for each intersection according to the urban scenario.

Urban scenarios may present consecutive intersections with a limited space between them. Then, a planning strategy is needed to find the best joint point between consecutive curves to generate a continuous path. Fig. 2 is an example of several consecutive intersections handled with three joined Bézier curves: C_1 (in orange), C_2 (in green) and C_3 (in red).

Different adjacent intersections have been handled in previous works [22]. However, they considered the previous (G_{n-1}) and the next (G_{n+1}) intersection points if the space between curves was lower than 20 meters. Doing this, the global control points maximum distance was fixed to the half of the distance of the minor shared segment, as it generated symmetrical curves.

Meanwhile, in the proposed real-time planner the location of the joint point is changed taking into account the

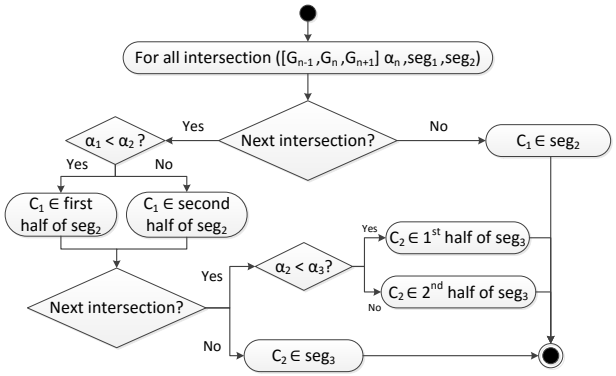

Fig. 3: Segmentation process of the consecutive intersections shared segment

next close intersections. That way, for each intersection the module plans the upcoming curve and the next one. To that end, the algorithm considers the angles of the three following curves and the length of the four segments.

There are two segments that are shared between the curves (see Fig. 2): seg_2 between the curves C_1 and C_2, whereas seg_3 is the segment between the curves C_2 and C_3. The principle that has been applied to divide the shared segments is to give more space to the more closed curve (i.e. the one with the least angle). According to the intersection angles, the more closed curve, the more segment stretch will be used for that curve.

In the example of Fig. 2, as the angle of the first intersection is smaller than the angle of the second intersection $(\alpha_1 < \alpha_2)$, for the first curve will be considered all the possible curves changing the external control point distance from the midpoint of the segment to the point G_n. The maximum value of this margin is limited by the maximum required spaces for performing the next curve in a proper way. In the same way, as the angle of the second intersection is greater than the angle of the third intersection $(\alpha_2 > \alpha_3)$, for the second curve will be considered the curves generated changing the external control point distance from the midpoint of the segment to the point G_n. If the point G_{n+1} were the path end, it would be considered the whole length of the segment (G_n, G_{n+1}), i.e., from 0 to 1.

The different curves changing the external control point distances are analyzed in order to optimize both the current and the following curve. In the example would be the curves C_1 in orange and C_2 in green for planning the first intersection, and the curves C_2 and C_3 for the second one.

Fig. 3 present all the cases that have to be evaluated, where the section of shared segment considered for placing the external control points in the curves is represented with an interval as a fraction of unity.

The algorithm developed to make the real-time planning is introduced in Algorithm 1. This algorithm plans the path of the current curve and the next in advance. Therefore, the way-points that define it are read (line 1, Algorithm 1). This

Algorithm 1 Real-time path planning algorithm

1: **Read** route way-points from the global planner
2: **Estimate** direction of rotation and angles of upcoming and two following intersections
3: **Calculate** intervals for the intersections shared segments
4: **Estimate** piece of the 1^{st} segment used for the curve
5: **Check** first, last or intermediate intersection
6: **for** the different 1^{st} curves moving the joint point on the 2^{nd} segment interval **do**
7: **Calculate** the piece of the 2^{nd} segment that can be used for the curve
8: **Estimate** the direction of rotation of the previous and following curve to load the curve from the right database
9: **for** the different 2^{nd} curves moving the joint point on the 2^{nd} segment interval **do**
10: **Estimate** the piece of the 3^{rd} segment that can be used to plan the 2nd curve
11: **Estimate** direction of rotation of previous and following curve to load the proper database
12: **Check** if both planned curves improve the optimal path
13: **Generate** the optimal upcoming curve

allows to calculate the direction of rotation and the angles of the upcoming intersections and the following two ones (line 2, Algorithm 1) planned in advance.

The intervals where the curves can be moved on the shared segments are calculated (line 3, Algorithm 1), following the criterion described before in Fig. 3. That way, the interval of segment where the consecutive curves can be joined is defined. This will allow to generate the different curves modifying the curve's beginning and ending points in order to find the optimal trajectory.

Line 4 describes the calculus of the piece of segment that can be considered for the upcoming curve. It considers whether it is the first curve (where the whole segment is used) or not (where it is used the rest of the segment that didn't use the previous curve). Since the database cannot consider neither all the possible angles nor all the possible distances for the segments, these values have to be approximated to the lower ones existing to load the nearest curve.

Line 5, the current intersection position (first, last or an intermediate one) is checked in order to generate the path for each curve loading the proper database for isolated intersections.

Later on, the location of the joint point of both upcoming and next curves is changed in the interval set before (line 4) to generate the different curves while moving its last point (line 6).

Once it is determined the piece of the second segment that is assigned to the upcoming curve (making the fitting as with the 1st segment, line 7), the direction of rotation of the following curves is analyzed in order to plan the upcoming curve (line 8).

In the same way as with the second segment, it is de-

termined the piece of the segment that can be considered to generate the different curves to plan in advance the following (second) one (line 9). As with the other segment, it is needed a fitting in order to load from the database the nearest curve (line 11). Previously is checked if there exist a third curve and its direction of rotation.

Once both upcoming (first) and next (second) curves are planned, it is checked if the path generated with both consecutive curves is the optimal one. This occurs when the sum of its finesses is lower than the local optimum (line 12). If so, the configuration of the curve is saved (longitudinal and lateral distances to the control points). Once it has finished the evaluation of all the different curves, the optimal one is generated for the upcoming intersection (line 13).

V. Results

This section presents the results to validate the real-time planner presented in comparison with the used in the team defined at [22]. To that end, the planning module has been developed in RTMaps[2], and the generated component has been tested with Pro-Sivic simulator[3]. This software simulates complex scenarios such as urban environments, allowing us to use the perception information and the models of the vehicles.

The experiments have been carried out on urban scenarios of multiple consecutive intersections, including ones with the same direction of rotation and others with the opposite. Fig. 4 represents the tested scenario. There, a lane road defined by squared intersections is assumed, where the lane width is 3 meters to fit with real scenarios. It is composed of two right turns and two left turns. Path planning considering pairs of curves was tested, taking into account a higher horizon of vision that considers the following curve and its respective turning angles and their direction of rotation. Upper part of Fig. 4 shows both the planned and tracked path with the previous team's implementation [22] and the proposed one. Whereas, in the lower part the curvature and its derivative are presented.

In the paths sub-figure the red continuous line shows the path generated with the previous team's implementation, detailed in [22], while the blue continuous line represents the one generated with the proposed planning system. The dotted lines represent the paths tracked by the vehicle during the simulation. The planned paths have been generated with 4th degree Bézier curves. This is the minimum degree in order to generate continuous enough paths to be joined for several curves. Meanwhile, in the lower sub-figure, the red and blue continuous lines represent the curvatures for both planning algorithms, the dashed lines represent their curvature derivative respectively.

Table II summarizes the results obtained. First, an improvement on the curvature is appreciated, being this clearly manifested in lower peak curvature values: 0.4295 m^{-1} is the maximum curvature with the previous team algorithm, whereas 0.2891 m^{-1} is the curvature with the proposed one. It means an improvement of a 32%. Furthermore, it can be also appreciated an improvement on the curvature derivative, being manifested in both peak and mean values. These improvements rely on the generation of smoother planned paths, allowing the vehicle to track more comfortable trajectories. This behaviour can be noticed on Fig. 4, where the path is better tracked by the vehicle in our approach based on the controller developed in [20].

[2]www.intempora.com

[3]www.civitec.com

VI. Conclusions and future works

In this work, a new real-time path planning algorithm has been presented for CTS in urban scenarios. Considering several intersections in a short period of time, an optimization in the path generation is developed considering an extended planning horizon: it plans the upcoming curve and the following one in advance for each intersection, trying to optimize both of them. Furthermore, an analysis of the division of the shared segment between two consecutive intersections has been done in order to evaluate the different curves moving the joint point between curves through the segment to generate a smoother and more continuous curvature on the path. The results obtained show a clear improvement in the path generation in terms of continuity, taking into account the path's curvature and its derivative. This let automated vehicles tracking smoother and more comfortable trajectories.

As future work, a comparison with other interpolating planning methods will be carried out. In addition, dynamic environments will be handled by introducing an algorithm to avoid obstacles in the path by considering the maneuver risk evaluating the obstacle's surrounding.

Acknowledgments

The authors want to thank the ANR-COCOVEA and ANR-VALET french projects for its support in the development of this work.

References

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

[2] A. Broggi, P. Cerri, M. Felisa, M. C. Laghi, L. Mazzei, and P. P. Porta, "The VisLab Intercontinental Autonomous Challenge: an extensive test for a platoon of intelligent vehicles," *International Journal of Vehicle Autonomous Systems*, vol. 10, no. 3, pp. 147–164, 2012.

[3] M. Parent, "Automated vehicles: autonomous or connected?" in *IEEE 14th International Conference on Mobile Data Management*, 2013.

[4] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, "Real-Time Trajectory Planning for Autonomous Urban Driving: Framework, Algorithms, and Verifications," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 2, pp. 740–753, 2016.

[5] T. Gu and J. M. Dolan, "On-Road Motion Planning for Autonomous Vehicles," in *Intelligent Robotics and Applications. 5th International Conference, ICIRA*, 2012, pp. 588–597.

[6] M. Fu, W. Song, Y. Yi, and M. Wang, "Path Planning and Decision Making for Autonomous Vehicle in Urban Environment," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 686–692.

[7] D. González and J. Pérez, "Control architecture for Cybernetic Transportation Systems in urban environments," in *IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 1119–1124.

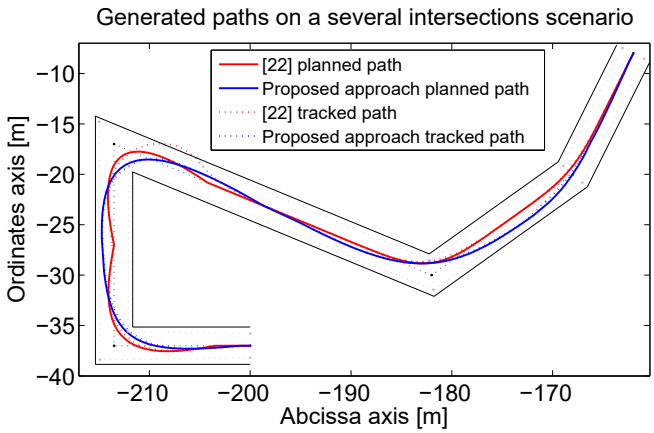

Fig. 4: Path planning over the tested urban scenario

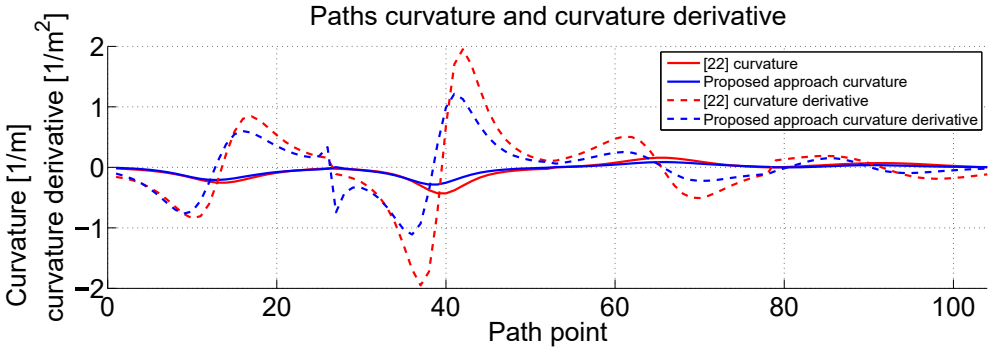

TABLE II: Experiments summary

Algorithm	**Measurements**			
	$\lvert\mu_k\rvert\ (1/m)$	$\lvert k_{max}\rvert\ (1/m)$	$\lvert\mu'_k\rvert\ (1/m^2)$	$\lvert k'_{max}\rvert\ (1/m^2)$
González et al., 2014	0.0929	0.4295	0.4286	1.9522
Proposed approach	0.0657	0.2891	0.2916	1.2554

[8] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, M. Panciroli, and A. Prioletti, "PROUD-Public road urban driverless test: Architecture and results," *IEEE Intelligent Vehicles Symposium (IV)*, 2014.

[9] T. Gu, J. M. Dolan, and J.-W. Lee, "On-Road Trajectory Planning for General Autonomous Driving with Enhanced Tunability," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*, 2014.

[10] F. Garrido, D. González, V. Milanés, J. Pérez, and F. Nashashibi, "Optimized trajectory planning for Cybernetic Transportation Systems," in *9th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2016, pp. 1–6.

[11] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, April 2016.

[12] D. Delling, P. Sanders, D. Schultes, and D. Wagner, *Engineering Route Planning Algorithms*. Springer-Verlag, 2009, ch. Algorithmics of Large and Complex Networks, pp. 117–139.

[13] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schrder, M. Thuy, M. Goebl, F. v. Hundelshausen, O. Pink, C. Frese, and C. Stiller, "Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, 2008.

[14] M. Pivtoraiko, R. A. Knepper, and A. Kelly , "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, 2009.

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

[16] H. Andreasson, A. Bouguerra, M. Cirillo, D. Dimitrov, D. Driankov, L. Karlsson, A. Lilienthal, F. Pecora, J. Saarinen, A. Sherikov, and T. Stoyanov, "Autonomous Transport Vehicles: Where We Are and What Is Missing," *Robotics Automation Magazine, IEEE*, 2015.

[17] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime Motion Planning using the RRT*," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[18] M. Brezak and I. Petrovic, "Real-time Approximation of Clothoids With Bounded Error for Path Planning Applications," *Robotics, IEEE Transactions on*, 2014.

[19] L. Han, H. Yashiro, H. T. N. Nejad, Q. H. Do, and S. Mita, "Bézier curve based path planning for autonomous vehicle in urban environment," in *IEEE Intelligent Vehicles Symposium*, 2010.

[20] J. Pérez Rastelli, R. Lattarulo, and F. Nashashibi, "Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles,"," in *IEEE Intelligent Vehicles Symposium (IV)*, 2014.

[21] INRIA, "CityMobil2 Vehicle Technical Specifications - Selection of offers for the city demonstrations," INRIA, Tech. Rep., 2014.

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