2023 31st Mediterranean Conference on Control and Automation (MED)
June 26 - 29, 2023. Limassol, Cyprus

2023 31st Mediterranean Conference on Control and Automation (MED) | 979-8-3503-1543-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/MED59994.2023.10185762

# Spline-Based Dynamic Object Handling in Autonomous Vehicles: A Model-Based Path Planning Algorithm

Aliki Stefanopoulou, Socratis Gkelios, Athanasios Ch. Kapoutsis, Elias B. Kosmatopoulos and Yiannis S. Boutalis

***Abstract*—In this study we propose a model-based dynamic path planning algorithm that is designed to navigate Autonomous Vehicles through complex and dynamic environments. To achieve that, a novel spline-based approach is utilized for the production of several candidate paths along a predetermined route and a Gaussian-based function is utilized for their evaluation. Our algorithm takes into account various factors, such as static and dynamic objects, to make the appropriate decisions for the vehicle's path, making it a promising solution for such objects during an autonomous vehicle navigation. The algorithm was tested in high-fidelity scenarios using CARLA Simulator, which is a powerful tool for simulating autonomous vehicle scenarios. The results indicate that the proposed algorithm is capable of generating efficient and safe paths for the vehicle to follow.**

## I. Introduction

In recent years, Autonomous Vehicles have been drawing a considerable amount of attention both from the academia and the industry. They are expected to have a significant impact in many different fields, such as transportation, industrial production and military, and are anticipated to minimize the contribution of humans in many daily activities. An Autonomous Vehicle is required to be able to navigate to a destination following an optimal global route, while planning reasonable local trajectories based on its real-time environment, that ensures the safety and the comfort of its passengers. This involves taking into account various factors such as the vehicle's current position and velocity, the layout of the surrounding environment, and any obstacles or constraints that may be present. The goal of path planning is to ensure that the vehicle arrives at its destination safely and in a timely manner, while also minimizing fuel consumption and maximizing overall efficiency.

This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH-CREATE-INNOVATE (project code:T2EDK-02743).

Aliki Stefanopoulou, Socratis Gkelios, Elias B. Kosmatopoulos and Yiannis S. Boutalis are with the Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece (alikstef1@ee.duth.gr, sgkelios@ee.duth.gr, kosmatop@ee.duth.gr and ybout@ee.duth.gr)

Athanasios Ch. Kapoutsis, is with Information Technologies Institute, The Centre for Research & Technology, Hellas, Thessaloniki, Greece (athakapo@iti.gr)

A commonly adopted approach for Autonomous Vehicles' architectures is to organize perception and decision-making tasks into a hierarchical structure. The perception system uses prior information and collected observation data to estimate the state of the vehicle and its environment, which is then used by the decision making system to control the vehicle and accomplish driving objectives. The decision making system is typically hierarchically decomposed into four components: route planning, behavioral layer, motion planning, and control system [1].

This study focuses on the decision making module of the autonomous vehicle, specifically the motion planning module that guides the vehicle to a destination point while considering static and dynamic objects in the environment.

One of the major challenges in path planning is dealing with uncertainty and dynamic objects, such as other vehicles, pedestrians, and obstacles that may move or change over time.

To address these challenges, the implemented algorithm focuses on three main parts: Given a predetermined global path that the autonomous vehicle needs to navigate in order to reach a destination point:

1) Generate a set of candidate paths based on the current location and environment of the autonomous vehicle,
2) evaluate the candidate paths based on static and dynamic safety and
3) select an optimal collision-free path to be followed.

Our algorithm makes use of a novel interpolation methodology that ensures driving safety and efficient handling of both static and dynamic obstacles.

Despite the presence of numerous interpolation-centric methodologies, the majority of these algorithms derive their feasibility from general assumptions that are based on limited empirical examination. As a result, the inferences drawn from such algorithms may not accurately reflect the complexities and difficulties inherent in real-world scenarios. This is particularly pertinent when it comes to handling dynamic obstacles, where it becomes challenging to simulate the intricate and unpredictable real-world setting from a "sterilized" environment that uses pre-determined cases that can be tackled with meticulous adjustments. In this context,

our contributions aim to address the aforementioned issues:

- We propose a novel dynamic object handling method by projecting the vehicle's dimensions through space considering their speed.
- We perform extensive simulations in realistic scenarios generated from CARLA Simulator that demonstrate the efficacy of our system.

## II. Literature Review

Many techniques used in navigation for AV's have been adapted from mobile robotics to address the complexities of road networks and driving regulations. These techniques can be divided into four main categories: **graph search**, **sampling**, **interpolating**, and **numerical optimization**. The most important algorithms used in motion planning for self-driving cars are described below.

### A. Graph search based planners

Automated driving involves finding a path from a starting point to a destination by traversing a state space, which is often represented as an occupancy grid or lattice that shows the positions of objects in the environment. Paths can be planned using graph searching algorithms, like Dijkstra's algorithm [2], and A-Star algorithm [3] and its variants, $D^*$ [4], the any-time dynamic $A^*$ ($AD^*$) [5], Field $D^*$ [6], $Theta^*$ [7], Anytime repairing $A^*$ ($ARA^*$) [7] and Anytime $D^*$ [8], among others.

These algorithms divide the configuration space into discrete cells and use a cost function to determine the best path. State lattice algorithm, which also used in automated driving, uses a discrete representation of the planning area with a grid of states. The path search in this algorithm is based on local queries and a cost function, and it can be implemented with different node search algorithms. The cost function decides the best path between the pre-computed lattices. A node search algorithm is applied in different implementations (e.g., ($A^*$) in [9] or ($A^*$) in [10]).

### B. Sampling Based Planners

These planners try to solve timing constrains, for example, planning in high dimensional spaces—that deterministic methods cannot meet. Probabilistic Roadmap Method (PRM) [11] and Rapidly-exploring Random Tree (RRT) [12] are sampling-based planners that solve timing constraints and can plan in high-dimensional spaces. They randomly sample the configuration space or state space to look for connectivity. The solutions are sub-optimal but fast and can handle non-holonomic constraints. RRT has been extensively tested for automated vehicles and is commonly used in on-line path planning. In [13] and [12] descriptions, applications and improvements of these algorithms are reviewed.

### C. Interpolating Curve Planners

Computer Aided Geometric Design (CAGD) is a technique often used to create smooth and efficient paths for a set of waypoints [14]. This method allows motion planners to generate a trajectory that is not only feasible but also comfortable and dynamic by taking into account various parameters. The process of creating new points within the range of existing reference points is known as interpolation. To generate a smoother path, these algorithms utilize a set of knots and consider factors such as continuity, vehicle limitations, and the dynamic environment [15]. In situations where obstacles are present, a new path can be generated to bypass them before returning to the original planned path. Various techniques of path smoothing and curve generation, such as interpolation, are widely used in the field of automated driving. In automated driving, different segments of a road network can be represented by combining known waypoints with straight and circular shapes. This is a simple method for creating a path for car-like vehicles [16], [17]. Clothoid curves are a type of curve that allow for smooth transitions between straight and curved segments, as their curvature changes linearly with their arc-length [14]. Polynomial curves [18], [19], [20], Bézier curves, and splines are all commonly used in automated driving for their ability to meet specific constraints and their low computational cost. Polynomial curves can be tailored to meet constraints at the beginning and end of a segment and are often used for line change and overtaking maneuvers. Bézier curves are defined by control points and are flexible and often used to approximate clothoid curves in automated vehicles [21], [22]. Splines are piecewise polynomial parametric curves that can be defined as polynomial curves [5], [18], b-splines [23], [24], or clothoid curves [25] and possess a high degree of smoothness constraint at the joint between the pieces of the spline.

### D. Numerical Optimization

Function optimization techniques are used to minimize or maximize a function while taking into account various constraints. They are commonly used in path planning to smooth out previously calculated trajectories [26], as well as to generate trajectories based on kinematic constraints [27]. It has been implemented to improve the potential field method in mobile robotics for obstacles and narrow passages and to find C2 continuous trajectories by minimizing a function that considers position, velocity, acceleration and jerk as planning parameters [27], [28].

These developments indicate that the foundation for automated driving has been established. Various designs use motion planners to ensure a safe and smooth path to follow. The current focus in the field is on addressing and mitigating potentially dangerous situations such as dynamic obstacle avoidance, emergency scenarios, negotiating intersections, and merging with other vehicles.

## III. Methodology

### A. Local Path Generation

*Center line construction*

Given a predetermined global waypoint list, which represents the High level route for the AV to reach its destination

point, a central line of the road is constructed, as depicted in Figure 1, using the method of cubic spline fitting. The constructed center line is created to ensure continuity and to calculate crucial information about the road network, such as the curvature of the road and the appropriate heading of the vehicle at each waypoint in the global path. By calculating the first and second derivatives of the curve, we generate a parametric cubic spline that represents the central line to be followed by the AV. Using the method of quadrature, we calculate the $arc$ lengths of all segments in the spline curve that is constructed by the global waypoint list. More precisely, the spline curve that describes the central line is parametrized as the $arc$ length of the curve, as described in [29]. Thus, the spline can be expressed as in Equation 1.

$$\begin{cases} x_0(s) = a_x s^3 + b_x s^2 + c_x s + d_x \\ y_0(s) = a_y s^3 + b_y s^2 + c_y s + d_y \end{cases} \quad (1)$$

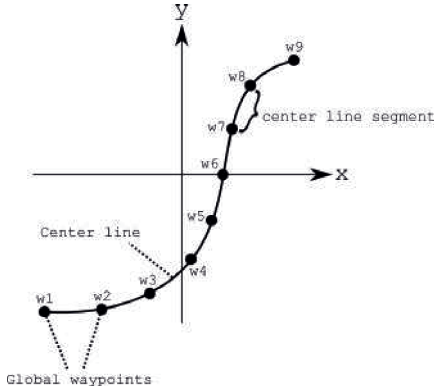



Fig. 1: Global waypoints and center line construction

The central line is divided in many segments, by the global waypoints. Each one of the segments generates a cubic spline as expressed in Equation 1, where $s$ is the $arc$ length of the segment, according to its previous waypoint in the global waypoint list and $(x_0, y_0)$ are the Cartesian coordinates of the point on the central line. In Equation 1, $a_x, b_x, c_x, d_x, a_y, b_y, c_y$ and $d_y$ are the coefficients. Through Equation 1, a position with $(x_0, y_0)$ coordinates can be expressed using the $arc$ length, $s$. According to [29], a point $p_0(s, \rho)$ is defined away from the road center at a given lateral offset $\rho$ from the center line. Thus, every point on the road and path can be transformed from Cartesian coordinates to $s-\rho$ coordinates. In the $s-\rho$ coordinate system, the heading $\theta_0$ and curvature $\kappa_0$ of every point on the central line can be computed using the first and second derivatives of a cubic spline with variable $s$, as described in Equations 2 and 3.

$$\theta_0 = \arctan \frac{dy_0}{dx_0} \quad (2)$$

$$\kappa_0 = \frac{x_0' y_0'' - x_0'' y_0'}{\sqrt{(x_0' + y_0')^3}} \quad (3)$$

where $x_0'$, $y_0'$, $x_0''$, $y_0''$ are the first and second derivatives of $x_0$ and $y_0$. Using Equations 1-3, the coordinates, heading and curvature of any point on the central line can be calculated.

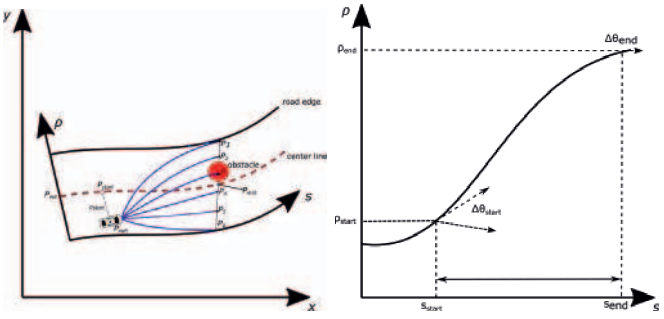
*Candidate paths generation*

Having defined the central line and translated all parameters to $s-\rho$ coordinate system, we are now capable of defining the candidate paths that will be considered at each time-step. Each set of candidate paths is generated with respect to the central line curvature. In order for the collision risk assessment to be as inclusive as possible, we take advantage of the Field of View and range of the sensors mounted on the Autonomous Vehicle and set the length of the generated paths, $\Delta l$ to the maximum length that the sensors can provide input of the environment. We take under consideration the total width of the available lanes to generate the candidate paths as described in Equation 4. The candidate paths are distributed evenly in the available width.

$$TotalWidth = width_{right} + width_{left} + c \quad (4)$$

where $width_{right}$ and $width_{left}$ are calculated according to the central line and represent the total available road width on either sides of the central line and $c$ is a constant.

As depicted in Figure 2(a), $P_{init}$ represents the first point of the center line, $P_{start}$ and $P_{end}$ are the projections of start and end points on the central line, respectively, for one step of planning. $P_{veh}$ is the current position of the vehicle. $P_1$ to $P_5$ are the end points of five path candidates, that are indicated by $c_1$ to $c_5$. As described in Equation 6, a function describing the relationship between the $arc$ length $s$ and offset $\rho$ is designed to provide a continuous change in the offset, which is an essential information, providing the AV's position according to the central line. In Figure 2(a), the $arc$ length $s_{start}$ and offset $\rho_{start}$ at the current position are given by finding the AV's position with respect to the central line. The relative angle $\Delta\theta_{start}$ is defined by the difference between the vehicle heading angle and tangent angle of the central line at the current position, as depicted in 2(b). Thus, the boundary conditions can be described as in Equation 5 [29].




(a) (b)

Fig. 2: (a) Construction of candidate local paths in the $(s-\rho)$ coordinate system. (b) Boundary condition in the Frenet $(s-\rho)$ coordinate system.

$$\rho(s) = \begin{cases} a(s-s_{start})^3 + b(s-s_{start})^2 + \\ c(s-s_{start}) + \rho_{start}, \text{if } s \in [s_{start}, s_{end}] \\ \rho_{end} \quad , \text{otherwise} \end{cases} \tag{5}$$

$$\begin{gathered} \rho(s_{start}) = \rho_{start}, \rho(s_{end}) = \rho_{end} \\ \frac{d\rho}{ds}(s_{start}) = \tan\Delta\theta_{start}, \frac{d\rho}{ds}(s_{end}) = 0 \end{gathered} \tag{6}$$

The calculation of the conditions $s_{start}$, $\rho_{start}$ and $\Delta\theta_{start}$ for the current position is straightforward, using the Equations 5 and 6. Furthermore, given offset at the end point $\rho_{end}$ in Equation 6, a set of coefficients $a$, $b$, $c$ in Equation 5 can be obtained. Therefore, by just defining different $\rho_{end}$ points, we can derive the exact candidate path segments though the calculation of $a$, $b$ and $c$ parameters.

*Candidate paths conversion to the Cartesian coordinate system*

The mapping from Cartesian coordinates to Frenet coordinates is often a major computational problem in driving simulators. The key element in this conversion is to calculate the point closest to the central axis of the central line and convert it to a three-dimensional point expressed in Cartesian coordinates. In order to avoid this unnecessary conversion of all the objects lying beside the autonomous vehicle, we convert the candidate paths, that are generated in the $s-\rho$, to the Cartesian coordinate system. Path candidate points in the Cartesian coordinate system can be derived with respect to the $arc$ length of the central line as in Equation 7 [29].

$$\begin{aligned} \frac{dx}{ds} &= Acos\theta \\ \frac{dy}{ds} &= Asin\theta \\ \frac{d\theta}{ds} &= A\kappa \end{aligned} \tag{7}$$

where $\kappa$ is defined as the curvature of the path candidate points, and can be calculated as in Equation 8:

$$\kappa = \frac{B}{A}\left[\kappa_0 + \frac{(1-\rho\kappa_0)(\frac{d^2\rho}{ds^2}) + \kappa_0(\frac{d\rho}{ds})^2}{A^2}\right] \tag{8}$$

In 7, $A$ and $B$ are

$$\begin{aligned} A &= \sqrt{(\frac{d\rho}{ds})^2 + (1-\rho\kappa_0)^2} \\ B &= sgn(1-\rho\kappa_0) \end{aligned} \tag{9}$$

The Cartesian coordinates $(x, y)$ and heading angle $\theta$ of the path candidate points can be calculated using the Equations 8 and 9.

### B. Cost Function

Having defined the candidate paths we now proceed with the evaluation and ranking of these candidate paths, considering their static and dynamic safety. A total cost function that consists of two separate cost functions that measure static safety and dynamic safety as described in Equation 10.

$$C_i = c_1 S_i + c_2 D_i \tag{10}$$

where $S$ is the Static safety cost function, $D$ is the Dynamic safety cost function and $c_1, c_2$ are weighs that give more or less priority to one term over the other.

The new selected path is chosen as the one that achieves the lowest score in the previously defined cost function, as described in Equation 11.

$$SP = \underset{i=1,2,\ldots,n}{\arg\min}\ C_i \tag{11}$$

where $SP$ represents the Selected Path and $n$ is the total number of the generated candidate paths.

*Static safety assessment* After converting all candidate paths into points with $x, y$ coordinates, we use the overlapping rectangles method, as described in Equation 12, to check for potential collisions, by projecting the autonomous vehicle's bounding box onto each point.

$$\begin{cases} R1_{topright}.x < R2_{bottomleft}.x \\ R1_{bottomleft}.x < R2_{topright}.x \\ R1_{topright}.y < R2_{bottomleft}.y \\ R1_{bottomleft}.y < R2_{topright}.y \end{cases} \tag{12}$$

If any of the cases in Equation 12 evaluate to true, it is assumed that following that path will lead to a collision with a static obstacle. The rectangles for the AV and obstacles are defined using their bounding boxes, as depicted in Figure 3. If this condition is met for any point on a candidate path, the path is flagged as occupied. This process is carried out using the Cartesian coordinate system to avoid the need for converting the obstacle's bounding box location to the Frenet $(s-\rho)$ coordinate system.

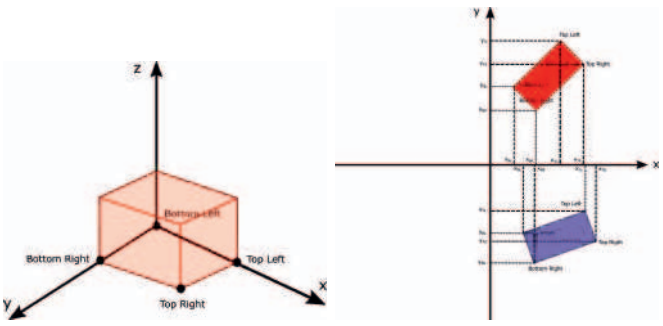



Fig. 3: Conversion from a bounding box to a corresponding rectangle

*Dynamic safety assessment* We consider as dynamic all the objects within a constant radius from the AV that move

with a speed greater than zero, including both vehicles and pedestrians. To asses possible collisions with dynamic objects, we use the same method of overlapping rectangles as in section Static safety cost function, but the projection of the AV to the candidate paths is calculated over time. Also, we follow the same method of projection for each one of the obstacles that lie within the aforementioned radius. The first step of this assessment is to calculate the time $t_i, j$ that is takes for the AV to reach each one of the points in the candidate paths, assuming that it is moving with its current speed, $u_{av,current}$, as described in Equation 13.

$$t_{p_{i,j}} = \frac{dist_{p_{i,j}}}{u_{av,current}} \tag{13}$$

where $dist_{p_{i,j}}$ is the distance from the AV to the point $p_{i,j}$ and $j = 1, 3, ..., m$ is the total number of points generated for each one of the paths. After calculating the time it takes for the AV to reach each one of these points, we project its bounding box, assuming that the center of the vehicle will be at location $(x = p_{i,j}(x), y = p_{i,j}(y), z = 0)$, and derive the corresponding rectangles of the AV at these positions, thus creating tuples with coordinates and time that the AV will be at these coordinates, as described in Equation 14 and depicted in Figure 4.

$$\begin{cases} position_{av,i,j} = (x = p_{i,j}(x), y = p_{i,j}(y), z = 0) \\ t_{p_{i,j}} = \frac{dist_{p_{i,j}}}{u_{av,current}} \end{cases} \tag{14}$$

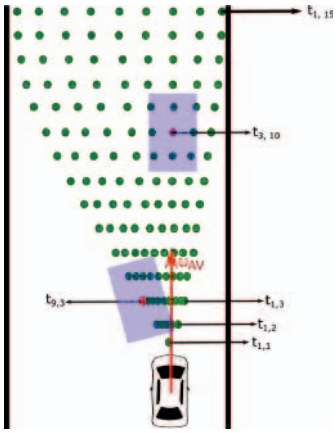



Fig. 4: Projection of the autonomous vehicle in the candidate paths through time.

By the same logic and taking under consideration the speed of a dynamic object, we can calculate its position at times $t_{i,j}$ for each point of each of the candidate paths, using Equation 14. We assume that the object will be moving straight and project it accordingly, as depicted in Figure 5. From each one of the aforementioned projections, we calculate the corresponding rectangle. Finally, we check for possible collisions through time, using the method of overlapping rectangles, as described in section III-B. If a path does not cross any dynamic object the collision check is defined as 0. If a path crosses a dynamic object, the collision check is defined as the highest value involved. The collision checks for dynamic objects are depicted in Figure 6(b).

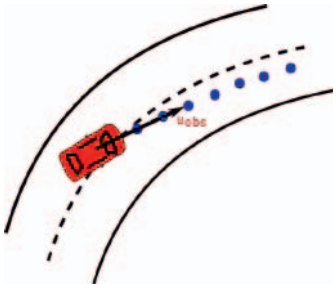



Fig. 5: Dynamic vehicle path projection through time.

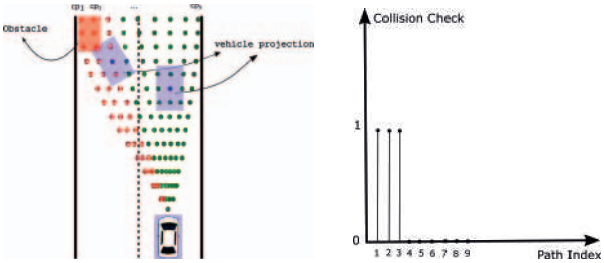



Fig. 6: Collision Check results.

*Collision risk assessment* The outcome of the collision check is crucial for path selection, but no other information is utilized, it may result in hazardous accidents. For instance, in Figure 6, if the criterion for path selection is the shortest path with no collision, the path $cp_4$ may seem like the optimal choice (as shown in Figure 6(a)). However, this path is dangerously close to an obstacle. To avoid collisions due to control errors or occluded obstacles, it's imperative to assess the risk associated with each path candidate. This study employs a method using discrete Gaussian convolution along with collision checks, as explained in Equations 15 and 16 to determine the risk for each path candidate.

$$f_s(cp_i) = \sum_{k=-N}^{N} g_i(k) R[k+i] \tag{15}$$

where $f_s(cp_i)$ is defined as the static safety cost function of path candidate $cp_i$ and $g_i(k)$ is the discrete inverted Gaussian function, which is defined as follows:

$$g_i(k) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(k-i)^2}{2\sigma^2}} \tag{16}$$

where $2N+1$ is the length of convolution, which is equal to the number of path candidates in our method, $R$ is the standard deviation of collision risk, which is a parameter for determining the effective range of collision detections.

## IV. Results and Analysis

The presented methodology was tested using predefined Autonomous Driving scenarios, using CARLA Simulator [30]. These scenarios included various roads with static and dynamic objects. The simulation results show that the proposed method can identify a safe path for navigating the vehicle and avoiding obstacles in most of the cases. More specifically, as depicted in Figure 7, the implemented algorithm is able to identify that following any of the generated candidate paths will lead to a collision with a static obstacle, and as a result, it stays still until an available local path is generated. In Figure 8, the algorithm predicts the bounding boxes of a pedestrian through time. Lastly, in Figure 9, slow leading vehicles result in possible collisions and the AV performs a lane change maneuver to evade them.

The results are presented in Table I. Scenario 1 and Scenario 2 presented small towns with a plethora of T-junctions, slow-moving vehicles and pedestrians. The roads in these two scenarios were single-laned, and as a result, the AV was not obligated to perform lane-changing maneuvers. The AV's average speed was 19.27 km/h in Scenario 1 and 11.9 km/h in Scenario 2. There were no collisions with either vehicles or pedestrians in any of these scenarios. Scenario 3 presented the most complexity among all tested scenarios, since it contained a 5-lane junction, a roundabout, unevenness and a tunnel. AV was obligated to perform lane-changing maneuvers in multiple occasions. The AV's average speed was 18.99 km/h. During testing, there were two collisions with moving vehicles (dynamic objects) and no collisions with pedestrians. Scenario 4 presented a infinite loop with a highway and a small town. The road network in this Town consisted of multi-laned roads. The AV's average speed was 44.57 km/h and the vehicle was obligated to perform lane changing maneuvers multiple times. In spite of the difficulties that this scenario presented, there were no collisions with either vehicles or pedestrians. Scenario 5 presented a Squared-grid town with cross junctions and a bridge. The road network contained multiple lanes per direction and as a result the autonomous vehicle had to perform lane changing maneuvers multiple times. Its average speed was 34.44 km/hour. During this scenario, the AV collided twice with a moving vehicle (dynamic object). Scenario 6 presented a town consisting of Long highways with many highway entrances and exits. The AV wad once again obligated to perform multiple lane changing maneuvers. During this scenario, its average speed was 30.44 km/h and no collisions were reported.

In the aggregate, according to Table I, the implemented algorithm managed to navigate the Autonomous vehicle for 2313 seconds (approximately 38.5 minutes) and 8483.46 meters (approximately 8.5 kilometers) and resulted in 4 collisions with vehicles in total and no collisions with pedestrians.

It is clear that the algorithm corresponds extremely well in cases of low moving speed in urban cities with high traffic. The majority of the collisions detected took place in cases the vehicle was moving with relatively high speed (more than 60 km/hour) and most of them occurred in cases that all of the evaluated candidate paths were unavailable, but the vehicle did not manage to reduce its speed in time.

| | Scenario 1 - Town01 | Scenario 2 - Town02 | Scenario 3 - Town03 | Scenario 4 - Town04 | Scenario 5 - Town05 | Scenario 6 - Town06 |
|---|---|---|---|---|---|---|
| Game Duration (Seconds) | 257.6 | 777.4 | 529.3 | 214.5 | 238.7 | 295.5 |
| System Duration (Seconds) | 354.78 | 1018.28 | 805.4 | 409.6 | 406.63 | 428.34 |
| Route Length (Meters) | 737.42 | 972.5 | 733.78 | 2628.59 | 1651.12 | 1760.05 |
| Collisions with layout | 0 | 0 | 0 | 0 | 0 | 0 |
| Collisions with Vehicles | 0 | 0 | 2 | 0 | 2 | 0 |
| Collisions with Pedestrians | 0 | 0 | 0 | 0 | 0 | 0 |
| Average Speed (km/h) | 19.27 | 11.9 | 18.99 | 44.57 | 34.44 | 30.44 |
| Maximum Speed (km/h) | 59.6 | 33.2 | 69.12 | 82.88 | 77.32 | 58.61 |

TABLE I: Experimental Results of the implemented algorithm - Meta Information

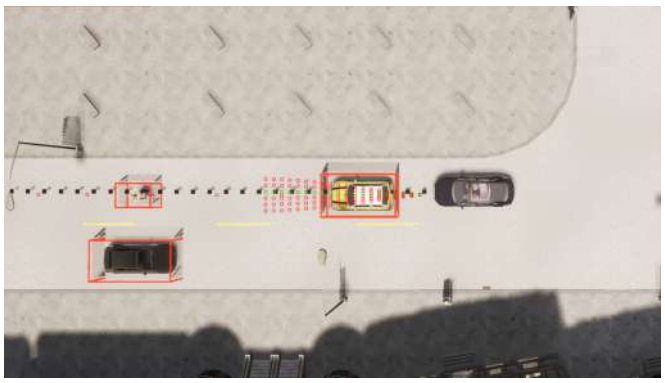
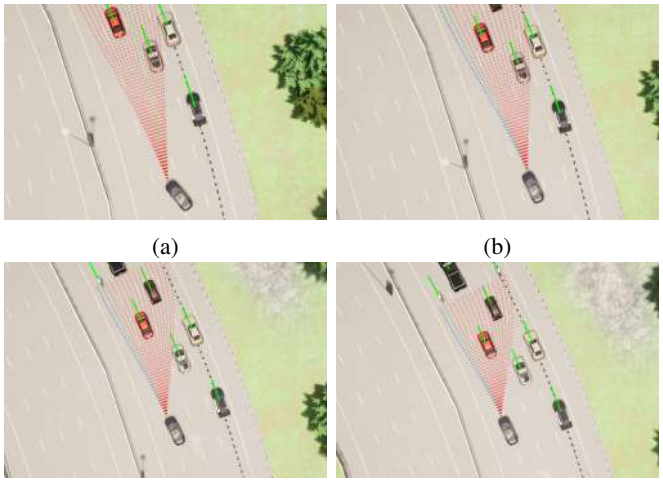


Fig. 7: All paths are blocked. The autonomous vehicle waits until it finds an available local path to follow.

Fig. 8: Prediction of the Bounding boxes of a pedestrian crossing the road.

## V. Conclusions and Future Work

In this study an integrated motion planning module that guides the vehicle to a destination point while considering static and dynamic objects in the environment was presented. The algorithm focuses on local path generation and the selection of the optimal collision-free path to be followed



(a) (b) (c) (d)

Fig. 9: (a) All candidate paths are blocked. The autonomous vehicle reduces its speed. (b) The evaluation of the candidate paths provides a collision-free path on an adjacent lane. (c) The autonomous vehicle performs a lane change maneuver. (d) Lane change maneuver has been performed. The autonomous vehicle has now evaded the slow moving vehicles in the first two lanes and is able to increase its speed.

along a predetermined global path. This algorithm is responsible for generating a set of path candidates in the Frenet coordinate system and selecting an optimal local path, making use of a novel cost function that considers static and dynamic object avoidance. This function prevents the autonomous vehicle from traveling unnecessarily close to static and dynamic objects and allows the vehicle to avoid static and dynamic objects, thus ensuring the safety of the passengers. Challenging scenarios (including various roads with static and dynamic obstacles) were used to test and evaluate the developed method. This algorithm demonstrated the capability to manage numerous static and dynamic obstacles simultaneously in contrast to similar techniques. The simulation results are promising and show that the proposed method can identify a safe path for guiding the vehicle and avoiding obstacles. In the future, the algorithm will be tested on a physical vehicle and will be evaluated in real-world scenarios. Future work will also contain a study of handling stochastic measurements and their effects on the implemented algorithm, a development of a behavioral layer for decision making and cost function expansion to account for comfortability of the AV's passengers.

## REFERENCES


[1] Brian Paden et al. "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles". In: *IEEE Transactions on Intelligent Vehicles* 1 (Apr. 2016). DOI: 10.1109/TIV.2016.2578706.

[2] F.M. Marchese. "Multiple Mobile Robots Path-Planning with MCA". In: *International Conference on Autonomic and Autonomous Systems (ICAS'06)*. 2006, pp. 56–56. DOI: 10.1109/ICAS.2006.38.

[3] Andrew Bacha et al. "Odin: Team VictorTango's entry in the DARPA Urban Challenge". In: *J. Field Robotics* 25 (Jan. 2008), pp. 467–492. DOI: 10.1007/978-3-642-03991-1_4.

[4] Rahul Kala and Kevin Warwick. "Multi-Level Planning for Semi-autonomous Vehicles in Traffic Scenarios Based on Separation Maximization". In: *Journal of Intelligent and Robotic Systems* 72 (Dec. 2013). DOI: 10.1007/s10846-013-9817-7.

[5] Maxim Likhachev and Dave Ferguson. "Planning Long Dynamically-Feasible Maneuvers For Autonomous Vehicles". In: (June 2008).

[6] Changwen Zheng. "A Linear Interpolation-Based Algorithm for Path Planning and Replanning on Girds". In: *Advances in Linear Algebra and Matrix Theory* 02 (Jan. 2012), pp. 20–24. DOI: 10.4236/alamt.2012.22003.

[7] Kenny Daniel et al. "Theta*: Any-Angle Path Planning on Grids". In: *J. Artif. Intell. Res. (JAIR)* 39 (Jan. 2014). DOI: 10.1613/jair.2994.

[8] Maxim Likhachev et al. "Anytime search in dynamic graphs". In: *Artificial Intelligence* 172 (Sept. 2008), pp. 1613–1643. DOI: 10.1016/j.artint.2007.11.009.

[9] Aleksandr Kushleyev and Maxim Likhachev. "Time-bounded lattice for efficient planning in dynamic environments". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 1662–1668. DOI: 10.1109/ROBOT.2009.5152860.

[10] Martin Rufli and Roland Siegwart. "On the Application of the D* Search Algorithm to Time-Based Planning on Lattice Graphs." In: Jan. 2009, pp. 105–110.

[11] S.M. LaValle and J.J. Kuffner. "Randomized kinodynamic planning". In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. Vol. 1. 1999, 473–479 vol.1. DOI: 10.1109/ROBOT.1999.770022.

[12] Sertac Karaman and Emilio Frazzoli. "Sampling-based Motion Planning with Deterministic mu-Calculus Specifications". In: Dec. 2009, pp. 2222–2229. DOI: 10.1109/CDC.2009.5400278.

[13] L.E. Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580. DOI: 10.1109/70.508439.

[14] Mišel Brezak and Ivan Petrović. "Real-time Approximation of Clothoids With Bounded Error for Path Planning Applications". In: *IEEE Transactions on Robotics* 30.2 (2014), pp. 507–515. DOI: 10.1109/TRO.2013.2283928.

[15] Larissa Labakhua et al. "Smooth Trajectory Planning for Fully Automated Passengers Vehicles: Spline and

Clothoid Based Methods and Its Simulation". In: vol. 15. Jan. 2006, pp. 89–96. ISBN: 978-3-540-79141-6. DOI: 10.1007/978-3-540-79142-3_14.

[16] James A. Reeds and Larry A. Shepp. "OPTIMAL PATHS FOR A CAR THAT GOES BOTH FORWARDS AND BACKWARDS". In: *Pacific Journal of Mathematics* 145 (1990), pp. 367–393.

[17] John Horst and Anthony Barbera. "Trajectory generation for an on-road autonomous vehicle - art. no. 62302J". In: *Proc SPIE* (June 2006), pp. 82–. DOI: 10.1117/12.663643.

[18] Aurelio Piazzi et al. "Quintic G2-Splines for the Iterative Steering of Vision-Based Autonomous Vehicles". In: *Intelligent Transportation Systems, IEEE Transactions on* 3 (Apr. 2002), pp. 27–36. DOI: 10.1109/6979.994793.

[19] S. Glaser et al. "Maneuver-Based Trajectory Planning for Highly Autonomous Vehicles on Real Road With Traffic and Driver Interaction." In: *IEEE Transactions on Intelligent Transportation Systems* 11 (Jan. 2010), pp. 589–606.

[20] A. Simon and J.C. Becker. "Vehicle guidance for an autonomous vehicle". In: *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No.99TH8383)*. 1999, pp. 429–434. DOI: 10.1109/ITSC.1999.821096.

[21] L.Z Wang et al. "An approximation approach of the clothoid curve defined in the interval [0, $\pi$/2] and its offset by free-form curves". In: *Computer-Aided Design* 33 (Dec. 2001), pp. 1049–1058. DOI: 10.1016/S0010-4485(00)00142-1.

[22] J. Sánchez-Reyes and J.M. Chacón. "Polynomial approximation to clothoids via s-power series". In: *Computer-Aided Design* 35 (Dec. 2003), pp. 1305–1313. DOI: 10.1016/S0010-4485(03)00045-9.

[23] Z. Shiller and Y.-R. Gwo. "Dynamic motion planning of autonomous vehicles". In: *IEEE Transactions on Robotics and Automation* 7.2 (1991), pp. 241–249. DOI: 10.1109/70.75906.

[24] Tomas Berglund et al. "Planning Smooth and Obstacle-Avoiding B-Spline Paths for Autonomous Mining Vehicles". In: *IEEE Transactions on Automation Science and Engineering* 7.1 (2010), pp. 167–172. DOI: 10.1109/TASE.2009.2015886.

[25] D. Walton and D.s Meek. "Technical section: A controlled clothoid spline". In: *Computers and Graphics* 29 (June 2005), pp. 353–363. DOI: 10.1016/j.cag.2005.03.008.

[26] Dmitri Dolgov et al. "Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments". In: *I. J. Robotic Res.* 29 (Apr. 2010), pp. 485–501. DOI: 10.1177/0278364909359210.

[27] Julius Ziegler et al. "Making Bertha Drive—An Autonomous Journey on a Historic Route". In: *IEEE Intelligent Transportation Systems Magazine* 6.2 (2014), pp. 8–20. DOI: 10.1109/MITS.2014.2306552.

[28] Julius Ziegler et al. "Trajectory planning for Bertha — A local, continuous method". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. 2014, pp. 450–457. DOI: 10.1109/IVS.2014.6856581.

[29] Xuemin Hu et al. "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles". In: *Mechanical Systems and Signal Processing* 100 (2018), pp. 482–500. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2017.07.019. URL: https://www.sciencedirect.com/science/article/pii/S0888327017303825.

[30] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator". In: *CoRR* abs/1711.03938 (2017). arXiv: 1711.03938. URL: http://arxiv.org/abs/1711.03938.