Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 1555 – 1570

doi:10.55730/1300-0632.3865

A new speed planning method based on predictive curvature calculation for autonomous driving

Bekir ÖZTÜRK[1], **Volkan SEZER**[2,*]

[1]Department of Mechatronics Engineering, İstanbul Technical University, İstanbul Technical University, İstanbul, Turkey

[2]Department of Control and Automation Engineering, Autonomous Mobility Group, İstanbul Technical University, İstanbul, Turkey

Received: 20.10.2021 • **Accepted/Published Online:** 08.03.2022 • **Final Version:** 31.05.2022

Abstract: As the number of vehicles in traffic is increasing day by day, the accident rates and driving effort are significantly raising. For this reason, ensuring safety and driving comfort is becoming more and more important. Driver assistance systems are the most common systems that are adopted for this purpose. With the development of technology, lane tracking support and adaptive cruise control systems are now being sold as standard equipment. More advanced research is being done for fully autonomous driving. One of the most critical parts of autonomous driving is speed profile planning. In this paper, a curvature-based predictive speed planner is designed using a fuzzy logic strategy. In addition to the predictive curvature value, lateral error to the planned path is considered for the final decision. With the help of the proposed predictive nature, which is not included in classical curvature-based planners, the vehicle is able to reduce the speed before cornering. Similarly, it starts increasing the speed when it is still in a curve but very near to the straight part of the road. In order to illustrate the efficiency of the proposed method and compare it with other approaches, the simulations are performed using realistic vehicle dynamics models of CarMaker software. After the comparative analysis, the designed planner is utilized on a real 1/10 scaled autonomous vehicle platform to show its real-world performance. The results show that the proposed speed planner is an effective and promising algorithm for both the comfort and path tracking performance.

Key words: Autonomous driving, speed planning, autonomous vehicles, fuzzy logic, robot operating system

1. Introduction

With the development in computation technology and the decrease of the sensor cost, the vehicles are getting more intelligent day by day. Driver assistance systems were first seen in commercial vehicles as electronic fuel control and cruise control (CC) in the 1980s. The driver can keep the vehicle at a certain speed due to CC which greatly reduces the driving efforts required by the driver. Driver assistance systems continue to develop rapidly and to increase the acceleration and braking performance of the vehicle such as the antilock braking system (ABS) in the 1990s. Autonomous vehicles can be widely used today due to systems such as lane tracking and adaptive cruise control (ACC) that were developed after the 2000s. Additionally, driving support systems that improve vehicle safety make the travel of passengers much safer.

Autonomous vehicles can mimic driver behavior observing their environment and following the designated path to the target location. Autonomous driving systems control the vehicle steering angle and gas-brake positions of the vehicle. For the development of autonomous vehicles, it is important to simulate the system in

*Correspondence: sezerv@itu.edu.tr

a virtual environment to save both time and cost. The difficulty of testing in a virtual environment is to achieve real vehicle outputs in the simulation. Vehicle modelling is highly significant to represent the real behavior of the vehicle. There are several tools that include reliable vehicle and environment models to obtain realistic simulation results. In this study, we use CarMaker[1] for this purpose.

Speed planning, which directly affects comfort and vehicle safety is a critical part of autonomous vehicles and robots. Dividing the local planning problem into two groups such as heading and speed planning is a common approach [1]. The speed planners in the literature are mostly designed by taking into account the comfort, handling [2] and obstacle avoidance performance [3–5]. In most urban driving scenarios, autonomous driving systems prefer soft speed profiles to keep passenger comfort high. Speed planners can be classified into two main groups such as; static and dynamic planners. Static speed planners represent the planning before the vehicle takes off. Dynamic speed planners are the planners that change the vehicle speed instantly while the vehicle is on the road [6]. In some applications, velocity planning can be done for fuel consumption minimization such as in hybrid vehicles [7]. [8] uses the dynamic limits of the vehicle in velocity planning. In [9], the speed planner considers the curvature of the path, longitudinal acceleration, and jerk parameters. The speed planners designed in [10] and [11] use curvature value to define upper speed limit. The path is defined as a point array and the speed is adjusted relative to the slope of the closest point in [12]. These curvature-based methods use the curvature value of the single actual point on the planned path and there is no consideration of the future parts. On the other hand, some studies such as [13] mimics the human behavior in autonomous velocity planning. The experiments with drivers show that drivers tend to slow down while approaching a corner and speed up while exiting a corner, predictively [14].

This paper proposes a novel predictive speed planner design for autonomous driving by imitating the driver experience. In the proposed approach, a certain part of the path ahead of the vehicle is considered to calculate a curvature value and the speed is adjusted accordingly. In addition to the predictive curvature value, lateral error to the planned path is considered for the final decision. The predictive behavior of this method provides an optimized speed for the path which increases the path tracking performance and comfort significantly. The speed planner is designed in Matlab/Simulink environment with the CarMaker integration in order to get realistic simulation results. The pure pursuit method is used to manage the steering of the vehicle. Visualization software is designed in Unity Platform to examine the simulations more efficient. Regarding the favourable simulation results, the tests are established on a real 1/10 scaled autonomous vehicle platform to verify the validity of the fuzzy speed planner in a real-time application. Finally, the simulation and test results are analyzed to prove the validity and efficiency of the proposed predictive fuzzy speed planner.

The paper is organized as follows. Section 2 defines the vehicle model and the CarMaker implementation on the Simulink environment. Section 3 briefly describes the path tracking method. In Section 4, the designed fuzzy speed planner and parameters are explained. The track where the simulations are carried out, benchmark metrics, the program developed in Unity platform and analysis of simulations are explained in Section 5. Section 6 explains the real-world implementation by introducing the autonomous vehicle hardware and illustrating the performance results. Conclusion and future works are provided in Section 7.

2. Vehicle model

The vehicle is basically built on three axes as the longitudinal (x axis), the lateral (y axis) and the vertical axis (z axis). In order to examine the acceleration and braking of the vehicle along x axis, it is necessary to

[1]IPG Automotive (2021). IPG Automotive [online]. Website https://ipg-automotive.com [accessed 10 September 2021].

construct the longitudinal model of the vehicle and to examine the forces generated when the steering angle of the vehicle is changed, it is necessary to construct the lateral model of the vehicle [15]. For the simulation of the forces acting on the vehicle from the road, the forces in the vertical axis of should be considered. CarMaker software, which has realistic vehicle models, is used for simulations with a complete model of the environment.

The IPG CarMaker is a simulation software where a vehicle model can be tested for various scenarios. For this purpose, multiple criteria can be compromised and be adjoined in the simulation. These can be a detailed model structure, road patterns or a traffic plan. Within this context, the CarMaker integration and testing software offer a wide range of flexibility and adaptability. Other simulation tools such as Matlab or Simulink can be employed simultaneously with this software. The input and output parameters of the CarMaker model are shown in Figure 1. The vehicle model parameters used in the simulations belong to the Tesla Model S, which is readily available in CarMaker. Vehicle parameters are shown in Table 1.

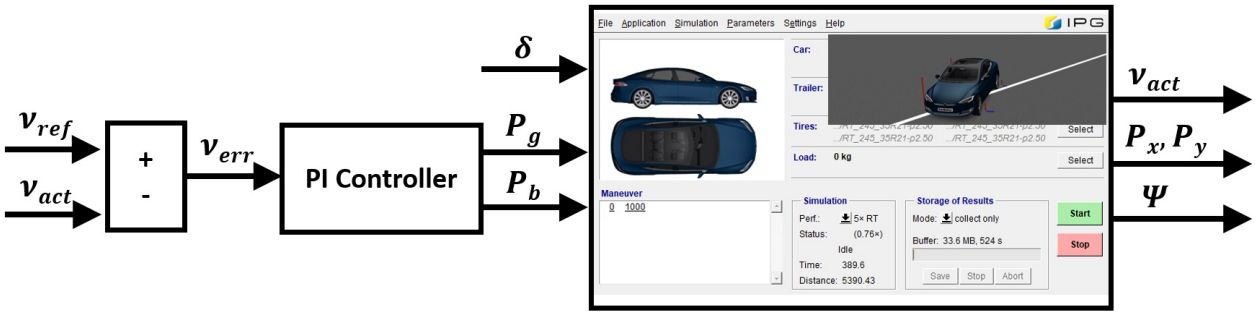

Figure 1. Inputs and outputs of CarMaker model.

Table 1. Vehicle parameters.

Parameter	Value
Vehicle mass [kg]	1840
Gear ratio	9.73
Effective wheel radius [m]	0.3
Moment of inertia [kgm^2]	1451
Distance between COG and front axle [m]	1.45
Distance between COG and rear axle [m]	1.555

Steering angle, gas and brake positions are taken as input parameters. On the other side, the output parameters consist of the position, orientation and speed of the vehicle. The PI controller is used to determine the gas and brake positions required for the vehicle to reach the desired speed. The coefficients of the PI controller is manually tuned (Kp = 0.02, Ki= 0.001) by observing the speed tracking performance and the same coefficients are used for all simulations.

3. Path tracking

Path tracking algorithms establish the required steering angle to keep the autonomous vehicle on the desired path. Various path tracking algorithms have been developed over time. The most common ones are the pure pursuit [16] and Stanley [17] methods. In this study, the pure pursuit method is selected for steering angle calculation which is briefly explained in this section. The main idea of the pure pursuit algorithm is calculating

the required steering angle geometrically in order to make the vehicle pass through the target point [16]. Destination point is the point located on the vehicle and facing away from the rear axle of the vehicle. This geometric relationship is shown in Figure 2. Steering can be calculated when the sine rule is applied as:

$$\delta(t) = atan\left(\frac{2Lsin(a(t))}{l_d}\right). \tag{1}$$

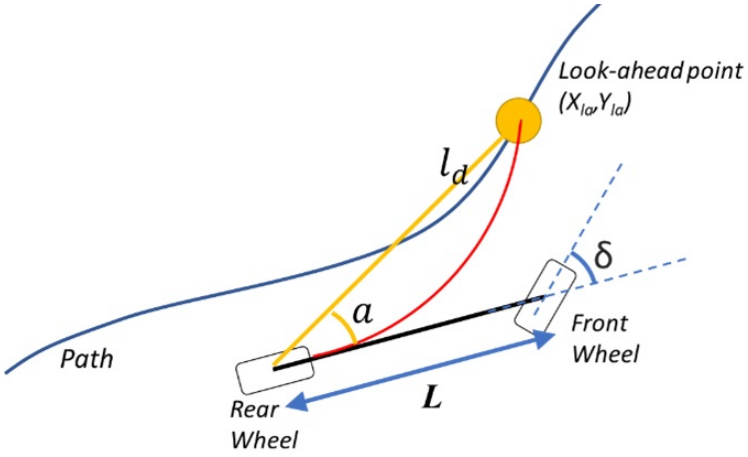

Figure 2. Pure pursuit geometric explanation.

In order to apply the pure pursuit algorithm to the model, the calculations are made in the following order:

1. Find the position and the orientation of the vehicle in global coordinates.
2. Find the closest point to the rear axle center of the vehicle on the road.
3. Find the target point (X_{la}, Y_{la}) on the road (where the distance between the rear axle of the vehicle and the target is equal to l_d).
4. Demean the target point from the global axis to the vehicle axis.
5. Calculate the steering angle using the steering angle equation. Go back to Step 1.

An important parameter of the pure pursuit algorithm is the l_d parameter that defines the look ahead distance. If this amount is kept smaller than the optimum value, the vehicle's steering wheel angle results with frequent and sudden changes. This behavior causes the vehicle to oscillate. If the amount of look ahead distance is greater than the optimum value, the vehicle's behavior will be smoother, but the road following performance falls down[2]. In order to provide a smooth ride and to efficiently track the desired path, the look ahead distance needs to be set carefully.

A new method is proposed in [18] that selects a look-ahead point heuristically by taking the relation between a vehicle and a path into consideration. This method becomes very advantageous for the vehicle to be able to converge to the desired path stably and track the path without engaging the cutting-corner problem. Based on the results of the simulations using this method, the vehicle is able to track the desired path more precisely.

[2]MathWorks Inc. (2021). MATLAB [online]. Website https://www.mathworks.com/help//robotics/ug/pure-pursuit-controller.html [accessed 10 September 2021].

4. Speed planning

The speed planner has a critical role in autonomous driving applications. The problem is defined as obtaining an efficient speed plan in order to track the planned path with minimum error while providing high comfort and low travel time. The requirements are conflicting (e.g., low travel time decreases comfort) and a new approach based on predictive curvature calculation is proposed for the solution in the next sections.

4.1. Design of predictive fuzzy speed planner

As it is explained in Section 1, the curvature-based approaches use the actual curvature value of the path for speed planning. On the other hand, we propose a curvature-based method that is inspired by experienced driver behavior. While driving on the road, experienced drivers slow down the vehicle when coming across a corner to be able to take the corner safely. Moreover, they increase the speed just before the end of the curve to decrease the travel time. Additionally, all drivers consider their lateral error on the path for adjusting the vehicle speed. Based on these observations, a fuzzy speed planner is proposed considering both the lateral distance of the vehicle to the road and the amount of predictive curvature value in front of the vehicle. Figure 3 illustrates the structure of the designed fuzzy speed planner. The average curvature of the road and the lateral distance between the vehicle and the road, are defined as system inputs. The output of the fuzzy decision maker is the risk factor. If the risk factor is high, then the desire for slowing down the vehicle is also high.

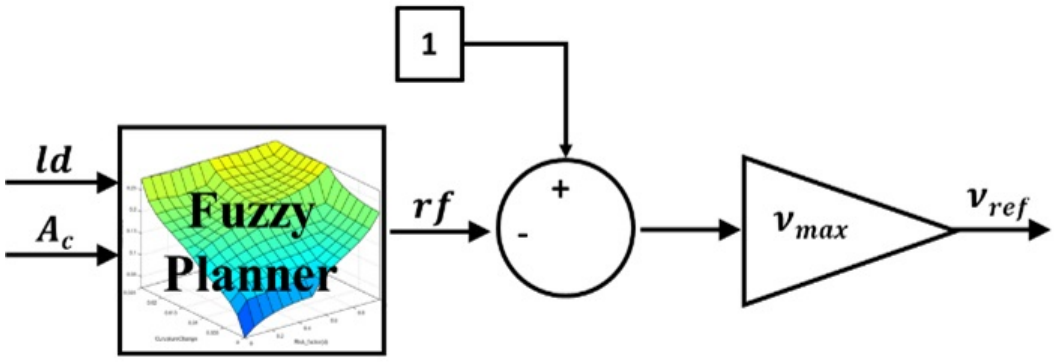

Figure 3. Designed fuzzy logic speed planner.

For the predictive curvature calculation, "n" number of points with "d" distance interval are considered. This means, speed is planned by looking up to (n-1)d meters in front of the vehicle. Curvature value C_i for each point is defined in Equation (2).

$$C_i = 1/R_i, \tag{2}$$

where R_i denotes the radius of the circle formed by the intersection of the point in the i^{th} index with its previous and the next point. For example, in Figure 4, the 2^{nd} and 4^{th} points are used for the radius information of the 3^{rd} point. C_{avg} is the average of "n" curvature values for each point ahead.

In order to calculate the average curvature value, it is necessary to have the curvature information of the points ahead along the path. To find the curvature of any point, the radius of the circle passing through three points (the point itself and its neighborhoods) must be found. The equation of the circle is described by Equation (3).

$$Ax^2 + Ay^2 + Bx + Cy + D = 0 \tag{3}$$

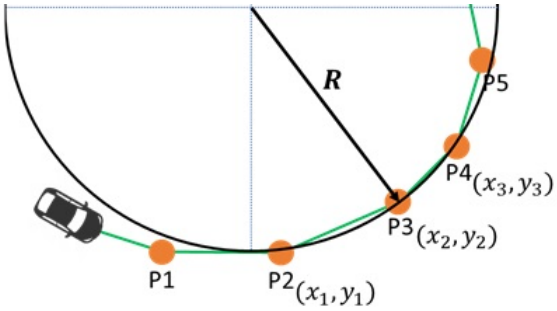

Figure 4. Curvature calculation.

To describe the circle passing through three points, a set of equations that can be defined by the determinant is obtained.

$$X = \begin{vmatrix} x^2+y^2 & x & y & 1 \\ {x_1}^2+{y_1}^2 & x_1 & y_1 & 1 \\ {x_2}^2+{y_2}^2 & x_2 & y_2 & 1 \\ {x_3}^2+{y_3}^2 & x_3 & y_3 & 1 \end{vmatrix} = 0 \tag{4}$$

If the determinant of Equation (4) is calculated with the help of the cofactor, the coefficients A, B, C and D are respectively equal to the determinants in the next equation.

$$X = (x^2+y^2)\cdot A + x\cdot B + y\cdot C + 1\cdot D$$
where
$$A = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}, B = -\begin{vmatrix} {x_1}^2+{y_1}^2 & y_1 & 1 \\ {x_2}^2+{y_2}^2 & y_2 & 1 \\ {x_3}^2+{y_3}^2 & y_3 & 1 \end{vmatrix}, C = \begin{vmatrix} {x_1}^2+{y_1}^2 & x_1 & 1 \\ {x_2}^2+{y_2}^2 & x_2 & 1 \\ {x_3}^2+{y_3}^2 & x_3 & 1 \end{vmatrix}, D = -\begin{vmatrix} {x_1}^2+{y_1}^2 & x_1 & y_1 \\ {x_2}^2+{y_2}^2 & x_2 & y_2 \\ {x_3}^2+{y_3}^2 & x_3 & y_3 \end{vmatrix}. \tag{5}$$

The $x_{1:3}$ and $y_{1:3}$ are the coordinates of the points which for the related circle. By using the coefficients in Equation (5), the radius of the circle passing through three points is calculated. Radius calculation is defined in Equation (6).

$$r = \sqrt{\frac{B^2+C^2-4AD}{4A^2}} \tag{6}$$

As an example, the average curvature values of the whole test track which is shown in Section 5.3, are illustrated in Figure 5.

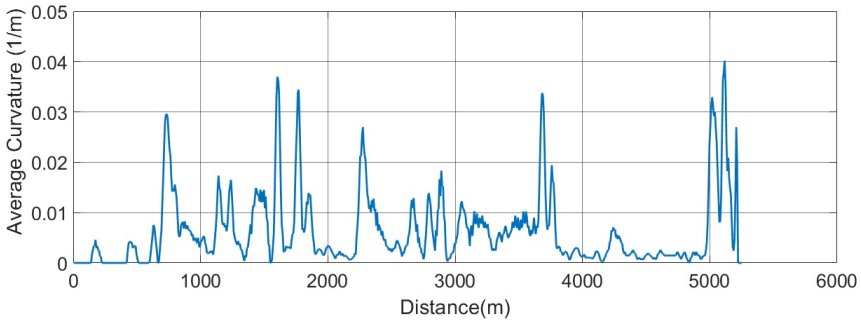

Figure 5. Average curvature values of the test track.

4.2. Fuzzy logic rules for the proposed method

The proposed speed planner is designed based on Mamdani-type fuzzy inference system using the "fuzzy logic toolbox" in Matlab. There are 3 membership functions for each input and 5 membership functions for the output as shown in Figure 6a. All inputs are normalized before entering the fuzzy block. The fuzzy surface after the designed membership functions and the rules is illustrated in Figure 6b.

As it is provided in [19], there is no exact method for choosing the membership functions. The shape of MFs depends on how one believes in a given linguistic variable. In this paper, triangular membership functions used since they are enough to reflect the definitions, have the advantage of simplicity and computational time.

Fuzzy rules are defined from human expertise, which makes the system fuzzy expert system (FES)[20]. Drivers always look ahead and make a predictive plan by analyzing the shape of the road (curvature) and their vehicle's position in the lane. As a driver for safe driving, if the lateral distance of the vehicle is low and the curvature ahead is low (straight road), then they increase the vehicle velocity. Similarly, when the lateral distance is high, or the curvature ahead is high, then they plan for decreasing the velocity. The rules in Table 2 are defined from this strategy. As it is illustrated in the table, the risk factor increases together with the higher values of curvature and the lateral error. This reduces the speed according to the system structure illustrated in Figure 3.

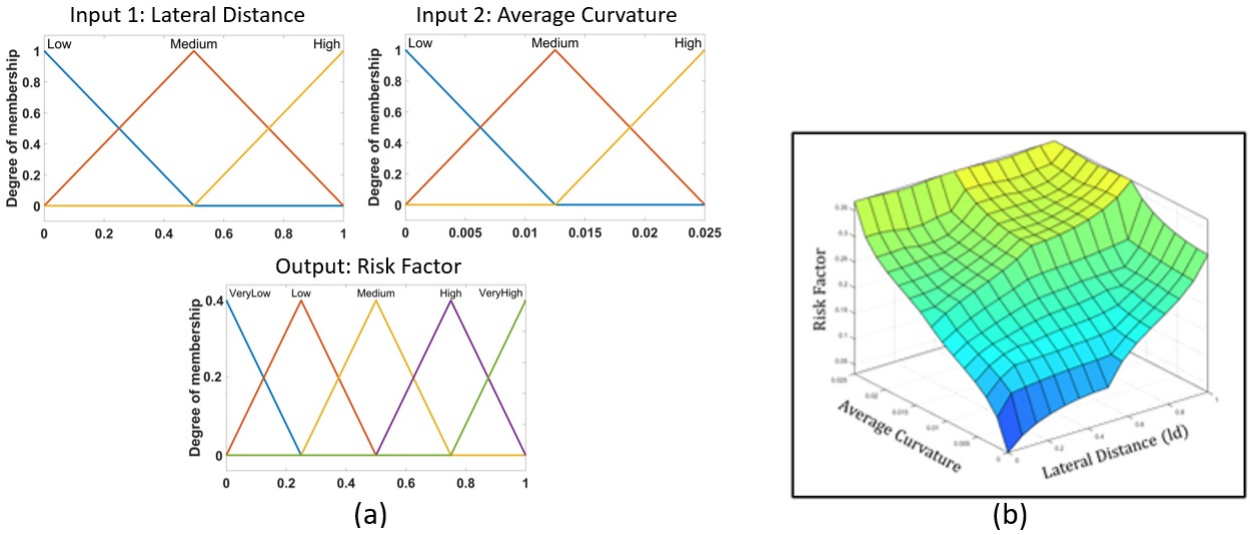

Figure 6. a. Fuzzy membership functions. b. Fuzzy rule surface.

Table 2. Fuzzy logic rule base (VL: very low, L: low, M: medium, H: high, VH: very high).

		Average	Curvature	Value
		L	M	H
Lateral Distance	L	VL	M	VH
	M	L	H	VH
	H	H	VH	VH

The upper limit of the lateral distance value is defined using road and vehicle widths. In modern highways, road width is 3.75 m [21] and vehicle width is taken as 1.8 m. In Figure 7, when the vehicle moves 1 m away

from the center of the lane, it is seen that it approaches the border of the lane. So, the upper limit value of the lateral distance is selected as 1 m.

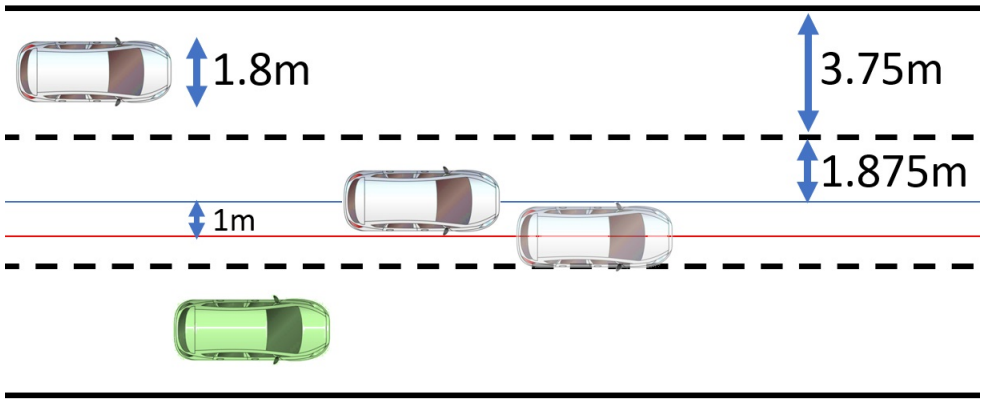

Figure 7. Relationship between road and car position.

The upper limit of the average curvature value is selected using the maximum vehicle cornering velocity. Vehicles need to slow down, such that they do not exceed a maximum centrifugal acceleration [22]. The relationship between maximum vehicle velocity and the road curvature is defined in Equation(7).

$$V_{max} < \sqrt{\mu_s R g} \tag{7}$$

Thus,

$$\frac{{V_{max}}^2}{\mu_s g} < R \tag{8}$$

Here, V_{max} is the maximum vehicle velocity, R is the curve radius, μ_s refers to the friction coefficient between road and tire, and g is the gravitational constant. In simulation maximum vehicle velocity is selected as 70 km/h (19.45 m/s). The friction coefficient for the dry asphalt is 0.9 [23]. So the minimum radius for turning without skidding can be found in Equation (9).

$$\frac{19.45^2}{0.9 \cdot 9.81} < R \Longrightarrow R > 42.84 \tag{9}$$

According to the relation between curvature and turning radius as provided in Equation (2), 42,84 m radius results with the curvature value of 0.023. For this reason, upper limit of the average curvature is selected as 0.025 in membership functions.

5. Simulations and analysis

In this section, the track where the simulations are performed, the developed software in Unity platform for analysis and the benchmark metrics for performance evaluation are explained. Turkish Grand Prix track which is a motor sports race track with 5.38 km length, is chosen for simulations. The track is chosen since it consists of sharp corners together with straight road segments and it is used for similar simulations such as in [24]. Using such kind of track, it is possible to examine the behavior of the autonomous vehicle both on straight and curved parts.

5.1. Benchmark metrics

In order to analyze the effect of the proposed speed planner, specific performance evaluation metrics are developed. The first of these is "tracking metric (TM)" which is designed for measuring the tracking performance based on vehicle's distance to the desired path (d_p). Second norm of the (d_p) lateral error values is calculated for obtaining TM metric as provided in Equation (10).

$$TM = \sum(|d_p|^2)^{1/2} \tag{10}$$

The second performance metric is the "comfort metric (CM)". The comfort metric is affected by the sudden maneuvers of the vehicle. The longitudinal forces acting on the vehicle chassis are caused by the change in the speed of the vehicle. Lateral forces are the combination of the change in the steering angle and the acceleration. Vertical forces and subsequent vibrations are related to the road disturbances.

The comfort metric is based on the resultant acceleration (aw) of the vehicle which is directly felt by the people during travel. The aw value is leveled in terms of comfort, according to the ISO standard [25]. As the acceleration value increases, the vehicle moves from the comfort zone to uncomfortable zone. Comfort levels based on ISO 2631-1 standard are provided in Table 3.

Table 3. ISO 2631-1 standard comfort levels.

Weighted acceleration (m/s^2)	ISO comfort level
<0.315	Not uncomfortable
0.315–0.63	A little uncomfortable
0.5–1	Fairly uncomfortable
0.8–1.6	Uncomfortable
1.25–2.5	Very uncomfortable
2	Extremely uncomfortable

The comfort metric is defined as the ratio of sum of the time intervals (shown as Δt_1 and Δt_2 in Figure 8) that the resultant acceleration aw pass beyond the limit (a_{lim}) to the total simulation time. The a_{lim} value is selected as $2m/s^2$ which defines "extremely uncomfortable" driving.

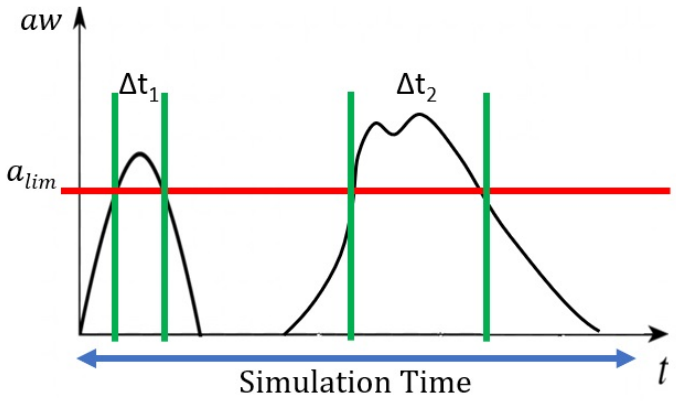

Figure 8. The resultant acceleration measured on the vehicle chassis.

The third metric is the travel time which is directly related to the average speed of the vehicle during simulation. The designed speed planner is compared with the speed planner which was created by considering only the lateral distance, ignoring the fixed speeds of 50, 60, 70 km/h and the average curve value of the designed speed planner. The target speed of the fuzzy speed planner and the lateral speed planner was entered as 60 and 70 km/h. The performance of the speed planner algorithms is calculated based on the comfort, the lateral distance to the road and the time to complete the runway.

5.2. Unity – car player

A car integrating and testing program is developed in the Unity environment [26] to simultaneously evaluate the values measured on the vehicle (such as the distance to the road, gas-brake positions) during the ride. Using the developed program, a great opportunity is achieved to examine data such as where the vehicle is at a specific time, the vehicle speed, the amount of curvature of the road etc. Figure 9 shows the graphical interface of the developed program.

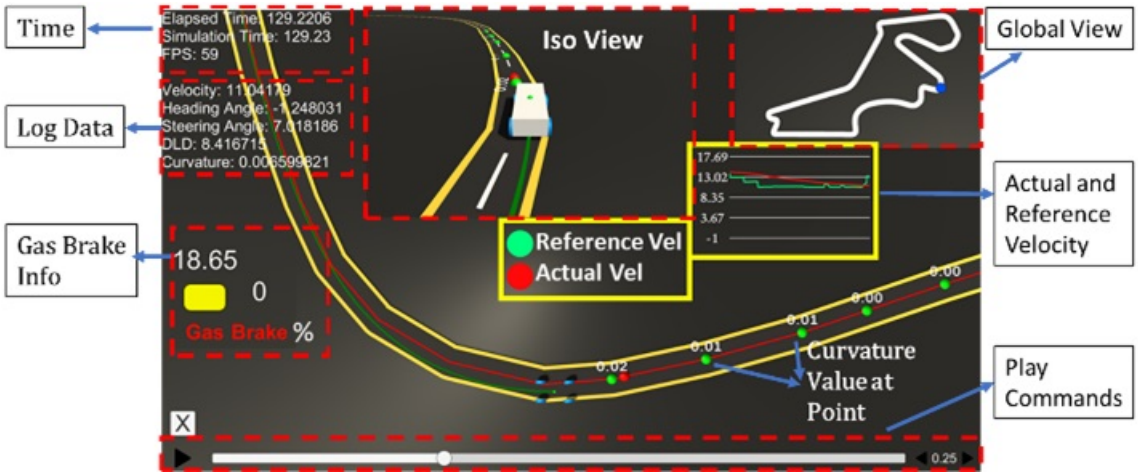

Figure 9. Unity – car player software gui.

5.3. Simulation results and analysis

The proposed predictive fuzzy speed planner's performance is directly related to the prediction distance for the calculation of curvature. In order to show this effect, we performed simulations with various number of points (1 to 7) to be considered for predictive curvature calculation as shown in Figure 10. The interval between points is selected as 7 m and maximum vehicle speed parameter V_{max} is set as 70 km/h. One extra simulation is made for comparison with classical curvature-based speed planning approaches, where the number of prediction points is 1 (no prediction) without considering the lateral error (green path in Figure 10). According to the results, it is observed (green and blue paths) that tracking performance increases when the lateral error is considered. Another result from this simulation is the effect of prediction distance. While considering shorter distance values decreasing the advantage coming from prediction, very long distances causes inclusion of unnecessarily far regions in the speed plan. According to the results, when the vehicle considers 5 points ahead (35 m), best tracking performance is obtained.

When the trajectory of the vehicle is analyzed, it is observed that the proposed speed planner slows down the vehicle, before entering the corner, as expected. Similarly, it starts increasing the speed even the vehicle

is still in a curvature part but the straight part is very close. These two predictive behaviors can be seen in Figure 11. The video of the fuzzy speed planner's performance prepared by the developed software based on Unity, can be watched from: https://youtu.be/hKWRgb-oGt4.

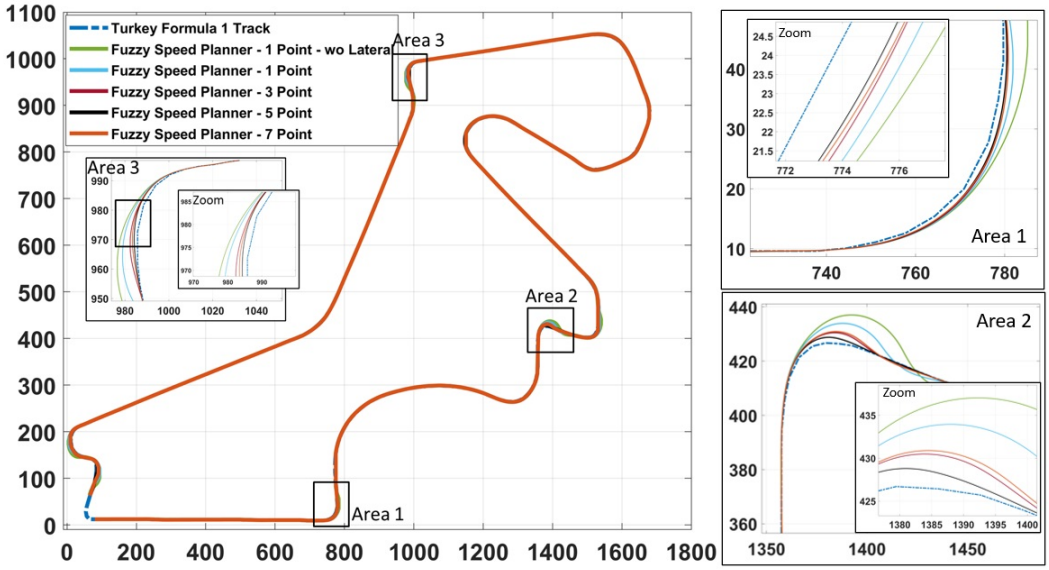

Figure 10. Simulation performance of different points into account in fuzzy speed planner.

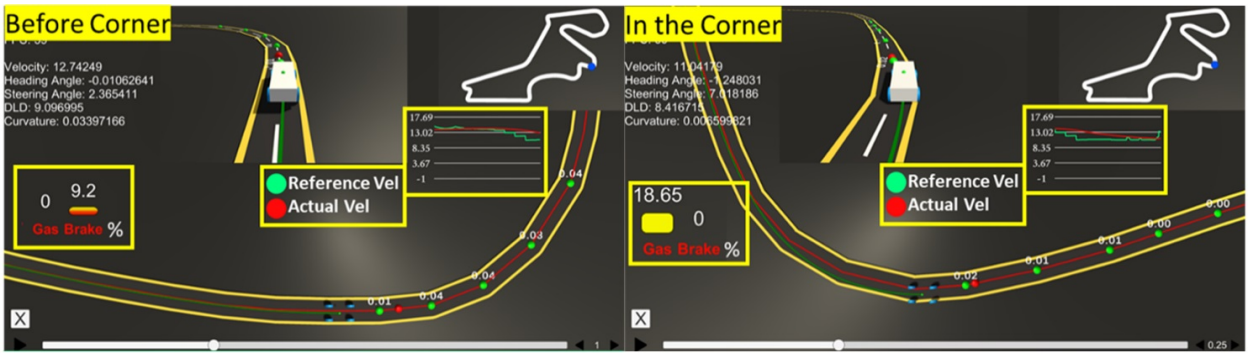

Figure 11. Unity – car player: vehicle gas-brake information.

In order to make an efficient comparison and analysis, several simulations are performed with different strategies and results are provided in Table 4. The first method called "fuzzy speed planner – 1P – wo Lateral" represents the classical nonpredictive curvature-based approach. It uses just the actual point's curvature without the lateral tracking error. The other fuzzy planners from 1 point to 7 point shows the performance of different prediction horizons. And finally, constant speed strategies such as 50 km/h, 60 km/h, and 70 km/h are illustrated in Table 4. According to the results, the proposed fuzzy planner with 5 prediction points results with the best performance in terms of tracking metric. Moreover, it is the second best in terms of comfort metric. The best comfort performance is obtained from 50 km/h constant speed strategy since it is the slowest one. For this reason, its travel time performance is the worst of all as expected. Figure 12 illustrates the paths of each strategy.

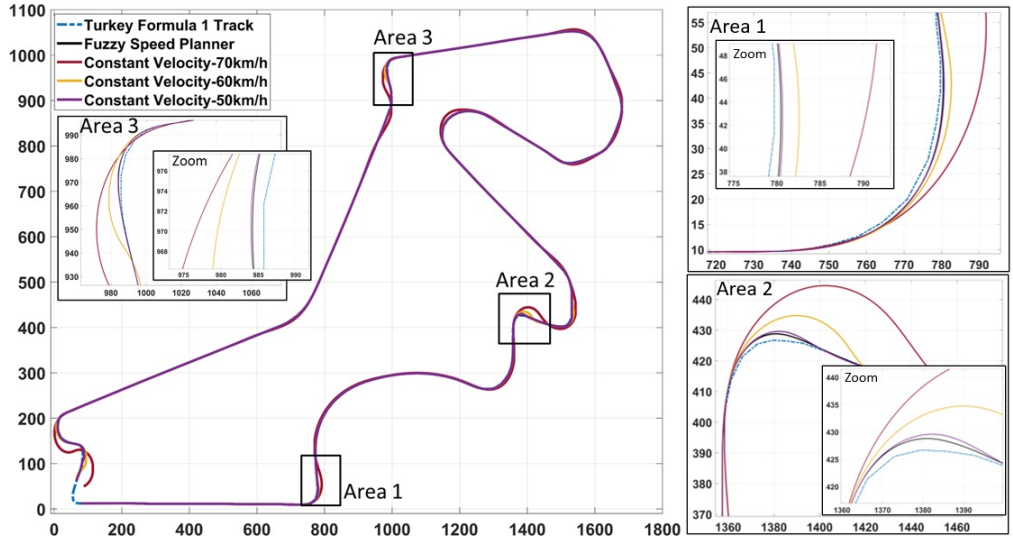

Figure 12. Simulation performance of different speed planners on the test track.

Table 4. Performance comparison of simulations.

Speed profile	Tracking metric (TM) (m)	Average speed (km/h)	Comfort metric (CM)	Travel time (s)
Fuzzy speed planner – 1P – wo lateral	285.42	64.35	0.325	302.1
Fuzzy speed planner – 1P	185.14	61.8	0.29	314.6
Fuzzy speed planner – 3P	84.02	59.8	0.256	325.1
Fuzzy speed planner – 5P	**65.82**	**58.15**	**0.245**	**334.3**
Fuzzy speed planner – 7P	99.81	57.24	0.231	339.6
Constant velocity - 70 km/h	876.72	68.09	0.368	285.5
Constant velocity - 60 km/h	207.25	59.54	0.275	326.5
Constant velocity - 50 km/h	68.57	49.94	0.178	389.3

6. Real-world implementation

In order to show the real time applicability of the propose planner in the real-world, it is implemented on a 1/10 scale RC car and compared with constant speeds. Different constant speeds have success on different performances. For example, while the path tracking the performance of low constant speed is good, the path completion time is bad. For real-world tests, a 1/10 scale Traxxas Slash 4x4 RC Car is used. Traxxas is driven by brushless DC electric motor with its Lithium Polymer battery system. This scaled autonomous vehicle can be seen in Figure 13a. Main components for autonomous driving are as follows:

- Nvidia jetson TX 2 is the mainboard of the RC car for autonomous driving algorithms.
- Slamtech A2 2D Lidar which is mainly used for mapping, localization and local perception.
- The SparkFun 9 IMU sensor is used as an auxiliary sensor to help the vehicle's position estimation.
- DC/DC Converter is used for providing required voltage supply for the sensors and the computer.

Figure 13b shows the grid map of the environment which is prepared by using simultaneous localization and mapping (SLAM) technique with the help of Gmapping library [27]. The vehicle's global path to be tracked on the map is shown in blue.

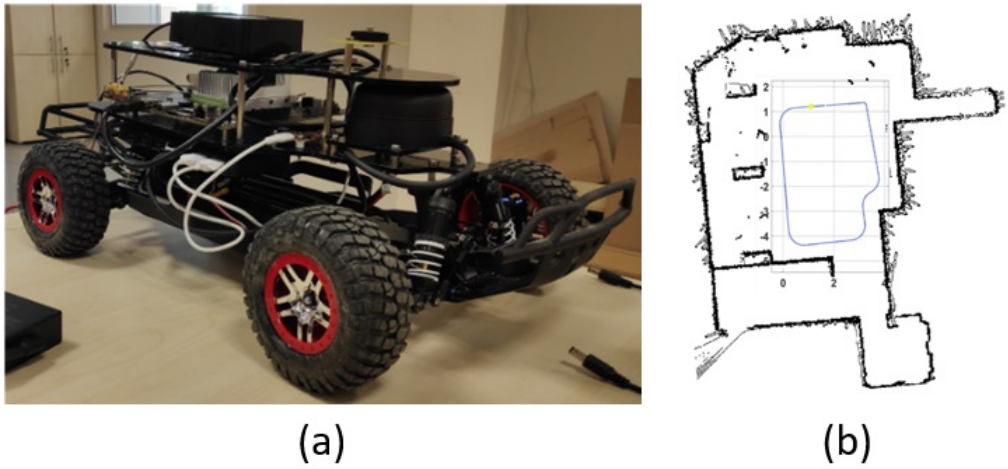

Figure 13. a. 1/10 Scaled autonomous vehicle. b. Grid map of the test environment.

The software development environment is located in the Ubuntu 18.04.5 Bionic Beaver operating system running on the Nvidia TX2 computer. All software architecture runs on the robot operating system (ROS). ROS is an open source metaoperating system [28]. It provides the services that is expected from an operating system, including hardware abstraction, low-level device control, implementation of commonly used functions, message passing between processes and package management. It also provides tools and libraries for writing and running codes on multiple computers. ROS continues growing its community and becoming widespread in the field of robotics day by day.

Figure 14 illustrates the designed ROS nodes for real application. In order to calculate the target speed using proposed method, path information, the closest point of the vehicle to the path and the position information of the vehicle are required. The path is defined as waypoints, prepared in CSV format and is rendered to be read by ROS. "Find nearest position" node is used to determine the closest point of the path to the vehicle. The path and the position information of the vehicle are required to find it. For this reason, "vehicle pose" node is developed, which uses adaptive Monte Carlo localization (AMCL) technique [29] to find the vehicle position on the map. The "find goal position and speed" node determines the point on the path which is as far as "look ahead distance" from the vehicle. It also calculates the target speed using proposed approach or classical methods. In order to implement fuzzy planners, scikit-fuzzy library is used in python environment [30]. "Pure pursuit" node calculates the required steering angle and sends it to the vehicle along with the target speed information.

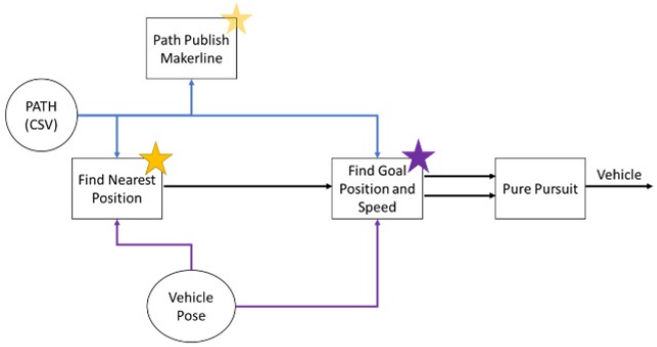

Figure 14. Designed ROS nodes.

Since the real platform is a scaled vehicle, V_{max} parameter is selected as 1 m/s. Five predictive points are used and the distance between each point is 7 cm. It is observed that the proposed method operates real-time without any problem. For comparisons, constant speeds are selected as 0.6 m/s, 0.8 m/s, and 1 m/s. Figure 15 illustrates the paths of each strategy. For simplicity, just 1 tour is visualized but each method is run 2 tours on the test track. The video of the fuzzy speed planner's real-world performance can be watched from: https://youtu.be/jrhpFG6ZKV0.

All results are provided in Table 5 for the real application. The proposed fuzzy planner has the best "tracking" and second best "travel time" performances. Best travel time performance belongs to the fastest (constant 1 m/s) strategy, but it loses tracking performance as the worst one. The proposed method has the second best performance in terms of "comfort" where the best comfort value comes from the slowest constant speed (0.6 m/s) strategy. On the other hand, this most comfortable strategy has the worst travel time performance as expected.

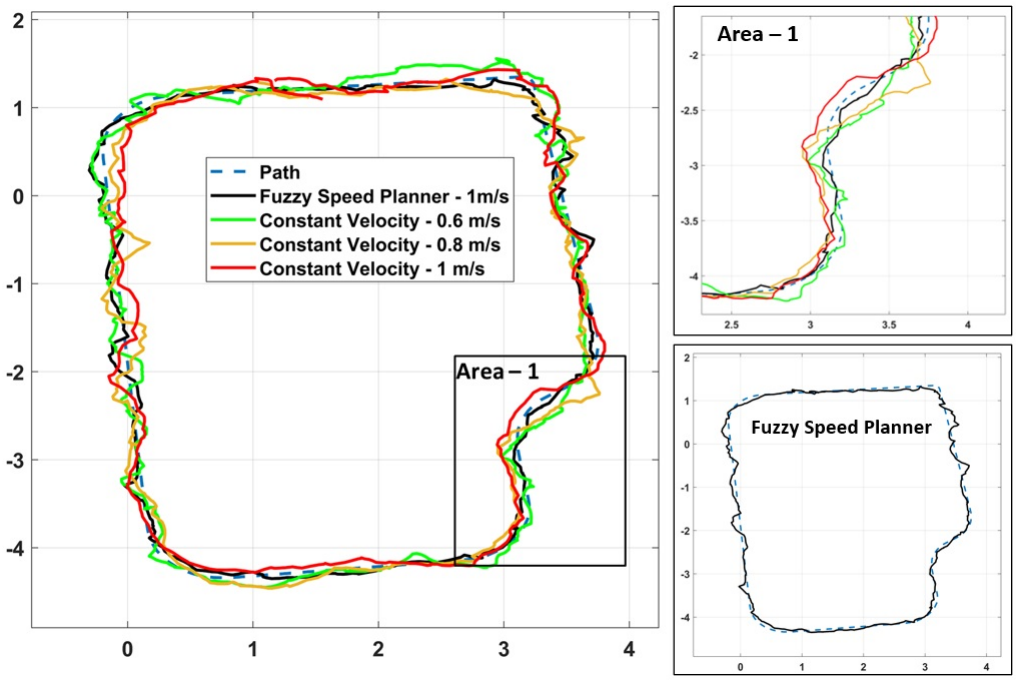

Figure 15. Performance of different speed planners on the test track.

Table 5. Performance comparison of real-world experiments.

Speed profile	Tracking metric (TM) (m)	Average vel. (m/s)	Comfort metric (CM)	Travel time (s)
Constant velocity - 0.6 m/s	2.334	0.654	0	51.91
Constant velocity - 0.8 m/s	2.609	0.862	0.087	39.409
Constant velocity - 1 m/s	2.815	1.095	0.122	31.016
Proposed fuzzy planner - 1 m/s	**1.738**	**0.906**	**0.055**	**37.488**

Although the proposed method does not give the best results for every metric, the efficiency of the approach is shown when all metrics are considered together.

7. Conclusion

In this paper, a new predictive speed planner is proposed and analyzed in simulations and real-world tests. The proposed speed planner calculates the curvature of the road ahead and combines it with the lateral path tracking error using fuzzy logic. This novel approach improves both the tracking and comfort performance comparing to classical curvature-based approach and the constant speed strategies. CarMaker software is used to make reliable simulations and comparisons. According to the simulation results, the proposed fuzzy planner with 5 prediction points results with the best performance in terms of tracking metric. Although the high performance on path tracking, comfort and travel time contradict each other, the new approach is the second best in terms of comfort metric without a dramatic increment on travel time. The efficiency of the proposed fuzzy speed planner is shown in a real application on a 1/10 scaled autonomous vehicle hardware. The results show that the proposed speed planner is an effective and a promising algorithm for speed planning. Real experiments can be improved for different types of scaled vehicles such as trucks or buses in further studies. The selection of the number of prediction points, interval between them, membership functions and the rules can be optimized according to the path as a future work. We use three levels (low/medium/high) for the inputs to get a more manageable rule table with a total of 9 rules. It has been seen that it is appropriate to have 5 levels for the output in order to express the scenarios with sufficient precision in 9 rules. It is always possible to increase the number of levels for inputs and outputs in a future study. Additional work can be done on the design of membership functions using tools such as artificial neural networks, for performance improvement.

References

[1] Perez-D'Arpino C, Medina-Melendez W, Guzman J, Fermin L, Fernandez-Lopez G. Fuzzy logic based speed planning for autonomous navigation under velocity field control. In: 2009 IEEE International Conference on Mechatronics. IEEE; 2009. pp. 1-6.

[2] González D, Milanés V, Pérez J, Nashashibi F. Speed profile generation based on quintic Bézier curves for enhanced passenger comfort. In: 2016 IEEE 19th international conference on intelligent transportation systems (ITSC). IEEE; 2016. pp. 814-819.

[3] Demir M, Sezer V. Design and implementation of a new speed planner for semiautonomous systems. Turkish Journal of Electrical Engineering & Computer Sciences. 2018;26 (2):693-706. doi: 10.3906/elk-1708-127

[4] Sezer V. Combined fuzzy approach for online speed planning and control with real vehicle implementation. International Journal of Vehicle Design. 2015;68 (4):329-345. doi: 10.1504/IJVD.2015.071098

[5] Sezer V, Ercan Z, Heceoglu H, Bogosyan S, Gokasan M. A new fuzzy speed planning method for safe navigation. In: 2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012). IEEE; 2012. pp. 381-386

[6] Zhang Y, Chen H, Waslander SL, Yang T, Zhang S et al. Toward a more complete, flexible, and safer speed planning for autonomous driving via convex optimization. Sensors. 2018;18 (7):2185. doi: 10.3390/s18072185

[7] Zhang B, Cao W, Shen T. Two-stage on-board optimization of merging velocity planning with energy management for HEVs. Control Theory and Technology. 2019;17 (4):335-345

[8] Herrmann T, Wischnewski A, Hermansdorfer L, Betz J, Lienkamp M. Real-time adaptive velocity optimization for autonomous electric cars at the limits of handling. IEEE Transactions on Intelligent Vehicles. 2020;6 (4):665-677.

[9] Villagra J, Milanés V, Pérez J, Godoy J. Smooth path and speed planning for an automated public transport vehicle. Robotics and Autonomous Systems. 2012;60 (2):252-265. doi: 10.1016/j.robot.2011.11.001

[10] Consolini L, Locatelli M, Minari A, Piazzi A. A linear-time algorithm for minimum-time velocity planning of autonomous vehicles. In: 2016 24th Mediterranean Conference on Control and Automation (MED). IEEE; 2016. pp. 490-495.

[11] Wang M, Liu Q, Zheng Y. A curvature-segmentation-based minimum time algorithm for autonomous vehicle velocity planning. Information Sciences. 2021;565:248-261.

[12] Nagel J, Trepagnier PG, Koutsougeras C, Kinney PM, Dooner M. The Culebra algorithm for path planning and obstacle avoidance in Kat-5. In: 2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06). IEEE; 2006. pp. 247-253.

[13] Cui Z, Guo X, Pei X. A Novel Velocity Planner for Autonomous Vehicle Considering Human Driver's Habits. SAE Technical Paper, 2020.

[14] Zhang D, Xiao Q, Wang J, Li K. Driver curve speed model and its application to ACC speed control in curved roads. International journal of automotive technology. 2013;14 (2):241-247. doi: 10.1007/s12239-013-0027-x

[15] Gillespie TD. Fundamentals of vehicle dynamics. SAE Technical Paper, 1992.

[16] Coulter RC. Implementation of the pure pursuit path tracking algorithm. Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[17] Hoffmann GM, Tomlin CJ, Montemerlo M, Thrun S. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In: 2007 American Control Conference. IEEE; 2007. pp. 2296-2301.

[18] Ahn J, Shin S, Kim M, Park J. Accurate path tracking by adjusting look-ahead point in pure pursuit method. International journal of automotive technology. 2021;22 (1):119-129. doi: 10.1007/s12239-021-0013-7

[19] Sadollah A. Introductory Chapter: Which Membership Function is Appropriate in Fuzzy System? In: Sadollah A, editor. Fuzzy Logic Based in Optimization Methods and Control Systems and Its Applications. Rijeka: IntechOpen; 2018. doi:10.5772/intechopen.79552

[20] Gilda K, Satarkar S. Review of fuzzy systems through various jargons of technology. Int J Emerg Technologies Innov Res. 2020;7:260-264.

[21] Teodorović D, Janić M. Chapter 11 - Transportation, Environment, and Society. In: Teodorović D, Janić M, editors. Transportation Engineering. Butterworth-Heinemann 2017; 719-858.

[22] Ni M. Study on Mechanical Effect of the Vehicle at Corners. In: 2015 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering. Atlantis Press; 2015; 614-617.

[23] Singh KB, Taheri S. Estimation of tire–road friction coefficient and its application in chassis control systems. Systems Science & Control Engineering. 2015;3 (1):39-61.

[24] Wang JY, Lin YB. Game ai: Simulating car racing game by applying pathfinding algorithms. International Journal of Machine Learning and Computing. 2012;2 (1):13. doi: 10.7763/IJMLC.2012.V2.82

[25] ISO I. 2631-1: Mechanical vibration and shock-evaluation of human exposure to whole-body vibration-Part 1: General requirements. Geneva, Switzerland: ISO. 1997.

[26] Murray JW. C# game programming cookbook for Unity 3D. CRC Press, 2021.

[27] Abdelrasoul Y, Saman ABSH, Sebastian P. A quantitative study of tuning ROS gmapping parameters and their effect on performing indoor 2D SLAM. In: 2016 2nd IEEE international symposium on robotics and manufacturing automation. IEEE; 2016. pp. 1-6.

[28] Quigley M, Conley K, Gerkey B, Faust J, Foote T et al. ROS: an open-source Robot Operating System. In: ICRA workshop on open source software. vol. 3. Kobe, Japan; 2009. pp. 5.

[29] Zou Q, Sun Q, Chen L, Nie B, Li Q. A Comparative Analysis of LiDAR SLAM-Based Indoor Navigation for Autonomous Vehicles. IEEE Transactions on Intelligent Transportation Systems. 2021:1-15. doi: 10.1109/TITS.2021.3063477

[30] Warner J, Sexauer J,Fuzzy S, twmeggs, alexsavio, Unnikrishnan A et al. JDWarner/scikit-fuzzy: Scikit-Fuzzy version 0.4.2. Zenodo; 2019.