IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 22, 2021, accepted May 17, 2021, date of publication May 28, 2021, date of current version June 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3084807

Efficient Local Navigation Approach for Autonomous Driving Vehicles

JOAQUÍN LÓPEZ[1], **PABLO SÁNCHEZ-VILARIÑO**[2], **RAFAEL SANZ**[1], **AND ENRIQUE PAZ**[1]
[1]Department of Systems Engineering and Automation, University of Vigo, 36310 Vigo, Spain
[2]Imatia Innovation S.L., 15008 A Coruña, Spain

Corresponding author: Joaquín López (joaquin@uvigo.es)

This work was supported by the Spanish Ministerio de Economía y Competitividad / Fondo Europeo de Desenvolvimiento Regional (MINECO/FEDER) through the Project SmartElderlyCar (CTAN) under Grant TRA2015-70501-C2-2-R.

ABSTRACT This paper presents an efficient and practical approach for a car navigation system (CVM-Car) based on the velocity space optimization paradigm. The method calculates the velocity control commands to keep the car in the lane while avoiding the obstacles detected by the proximity sensors. The car has to follow a road path consisting of a sequence of lanelets. This approach is a lower-level reactive control that combines the pure pursuit method to obtain a reference curvature and a reactive control algorithm that keeps the vehicle in the center of the lane' s free space while avoiding obstacles that can partially block it. CVM-Car formulates local obstacle avoidance as a constrained optimization problem in the velocity space of the car. In addition to the vehicle dynamics and obstacles constraints included by the curvature method, car-shape and non-holonomic restrictions are considered in the CVM-Car velocity space. The method has been applied to an autonomous vehicle prototype.

INDEX TERMS Autonomous vehicles (AVs), obstacle avoidance, reactive control, vehicle motion control.

I. INTRODUCTION

Academic and industrial research, into self-driving vehicles has steadily increased over recent decades due to the technology' s potential impact and social benefits. Reduced traffic fatalities, shorter commute times, improved road safety, increased comfort and efficiency, and greater mobility for more people are just some of those benefits.

Autonomous vehicle control systems must be able to take driving decisions based on their prior knowledge (road maps, traffic rules, sensor models and vehicle dynamics) and observations of the particular traffic situation. Those observations come from the perception system ([1]) that can include different on-board sensors, such as odometry, cameras, LIDARs, GPS units and SONARs. At the same time, the decisions are variables that control the vehicle' s motion.

Most implementations of these navigation systems are hierarchically broken down into at least three different components ([2]): Route planning, behavioral decision making and local navigation. Route planning computes the route on the road map to reach the goal. The route can include diverse scenarios such as different intersection types: yield, stop, merge, crosswalk, etc. For each scenario, the vehicle might encounter a variety of situations that must be detected by the perception system: pedestrians on the crosswalk, cars in the intersection, car approaching in the left lane, etc. Behavioral decision-making combines that route with information provided by the perception system to figure out the situation in the current scenario. According to the situation, a different motion goal is sent to the local navigation module. Motion goals typically take the form of discrete motion goals ([3]) such as driving along a road lane, stopping at a point before the intersection or maneuvering to a specific point in an open area.

In the solutions with a clear division between high-level planning and lower-level control, a path planner calculates the path according to the map while a local reactive navigation system concentrating on very short-term reasoning generates the next action for the vehicle ([4]; [5]). The route obtained by the global planner works as a reference for the local navigation system. Local navigation is performed in the control space to dynamically generate feasible actions. The research presented in this paper falls into this set of solutions. One major limitation of these purely local approaches is that the vehicle may get stuck in local minima. However, an executive layer in our approach detects local minima and

The associate editor coordinating the review of this manuscript and approving it for publication was Xiwang Dong.

uses the route planner to find an alternative [6]. Another drawback of these approaches is that they are unable to perform complex multi-stage maneuvers, such as the ones required in parking ([7]) but in our approach off-road and unstructured parking lot scenarios are managed as a separate case, as described in [8].

In this paper, we deal with the local navigation problem by implementing a new method (CVM-Car) based on the pure pursuit algorithm ([9]) and the Curvature Velocity Method (CVM) ([4]). The system has been implemented within an architecture that deals with the complex maneuvers and local minima. The first step (based on pure pursuit) obtains the curvature reference to follow the lane regardless of the obstacles. The curvature reference is the goal reference for the second step that handles obstacle avoidance while staying in the lane. The idea of using a curvature method for a car-like vehicle was also proposed in [10] using BCM instead of CVM. In that case the BCM curvature method was directly applied to a car-like vehicle resulting in a very restrictive method. In this paper, a new method based on curvature velocity is developed to take into account the shape and non-holonomic nature of a car-like vehicle.

Therefore, the main contribution of this paper is a novel local navigation algorithm for autonomous driving vehicles that is very fast (execution times below 2.0 milliseconds) and safe because of the curvature velocity restrictions. In addition, the maximum speed is adapted to the curvature of the road and the vehicle describes a smooth trajectory towards the center of the lane. The method has been implemented within our car navigation architecture and tested first in two different simulators V-REP ([11]) and CARLA ([12]) and finally in an autonomous vehicle prototype.

This paper is organized as follows. The next section introduces works related to this research. Section III defines the basics of the solution proposed for local navigation. Sections IV and V include a detailed description of the obstacle avoidance part of the algorithm. Tests to evaluate the performance of this solution and their results are presented in Section VI, which concludes the paper.

II. RELATED WORK

Local navigation involves reaching a motion goal safely, avoiding all the obstacles that appear during navigation. The motion goal is given in different ways according to the global navigation framework. For example, it can consist of a set of waypoints, a lane to follow, a point where the vehicle has to stop once reached, or even a global path. The problem is commonly solved using reactive algorithms, which at each control step decide the desired motion towards the target while accounting for the presence and expected motion of any perceived obstacles.

The most basic local navigation algorithms are path-tracking methods that do not avoid obstacles. These algorithms use vehicle position and odometry to control vehicle speed and steering to follow a specified path. A number of these local navigation approaches such as Follow-the-carrots, pure pursuit ([9]) and Follow-the-past have been described in the literature. In [13] the vehicle velocity is set according to the scenario and the steering is obtained as a function of lateral offset and heading offset with respect to the trajectory. These techniques have the advantages of needing, in general, low computational requirements and providing good performance, although they require obstacle-free trajectories.

Pure pursuit is probably the most common and one of the most robust and reliable geometric path tracking methods currently available. This approach and its adaptations have been employed for explicit path tracking in many vehicle navigation applications, including some of the vehicles that participated in competitions sponsored by the Defense Advanced Research Projects Agency (DARPA) for autonomous vehicles: the DARPA Grand Challenge and the DARPA Urban challenge.

Recently, some approaches have presented the navigation problem as finding the sequence of motions (trajectories) to reach the goal ([2]). For example, in [3] the motion planner tracks a path by generating a set of candidate trajectories that follow the path to varying degrees and then selecting the best trajectory from this set. The chosen trajectory is then directly executed by the vehicle according to an evaluation function. The trajectory in these cases is a set of feasible actions between initial and desired vehicle states computed using a predictive trajectory generator model [14]. In order to ensure no collisions with obstacles, some kind of convolution might need to be performed between the vehicle' s footprint and the local map for the different positions on the trajectory ([8]).

Once an obstacle-free local trajectory is obtained, a tracking algorithm can coordinate the steering, engine and brakes to follow the desired path ([15]). There are approaches that propose enhancements to the path tracking accuracy, some of which require a detailed system model such as those based on PID controllers. For example, [16] uses generalized gain scheduling and [17] adds Speed-based Acceleration Maps (SpAM) to the PID controller. Others, such as the Fuzzy logic controllers ([18]; [19]), do not require a detailed model even though they need a considerable number of vehicle tests for parameter calibration. Additional methods include a predictive control framework model like the one in [20] and [21].

A slightly different scheme is presented in [22], which uses state-space sampling-based trajectory planning to take obstacles into account. As in [3], the algorithm generates a set of candidate trajectories and selects one. The set of smooth and kinematically feasible paths are generated by using a model-based predictive path generation algorithm. The best trajectory is selected on the basis of an objective function that considers both safety and comfort performance. A collision test is performed to trim the trajectories that hit obstacles. To reduce computational complexity, the rectangular shape of the vehicle is approximated as a set of circles with the same radius that contains the vehicle. This avoids computationally complex convolutions. Even so, an occupancy grid map is used to represent the local environment perception information and the collision test has to be performed on each point of the trajectory. An obstacle space-time grid map was also used in [23] to define a velocity control strategy for collision avoidance in autonomous agricultural vehicles.

A hybrid method that independently handles the geometric constraints (obstacles) on one layer and the kinematic and nonholonomic constraints on another layer is shown in [24]. The method uses a search-based global path modification to deal with the geometry constraints and a multi-stage state sampling method to generate a kinematically feasible path. However, the search, and therefore optimization, is performed separately instead of by searching for an optimal trajectory for all the restrictions combined.

Another approach to the local navigation problem, widely used in the mobile robot field, is to use a local reactive obstacle avoidance method to avoid unexpected static and dynamic obstacles in a partially known environment. The most well-known methods from the extensive literature on local navigation approaches include Artificial Potential Fields, the Vector Field Histogram, Dynamic Window, and Curvature Velocity.

In the recent years, several solutions based on Model Predictive Controllers (MPC) have also been presented. Most of them are only aimed at following the path [25] but lately some of them include restrictions to avoid the obstacles [26], [27]. The performance obtained by these methods is very good but with high processing requirements since they need to integrate the obstacles in the map, obtain the restrictions and then execute the optimization process several steps ahead, up to the prediction horizon. Other solutions try to avoid or mitigate collisions using predictive occupancy maps [28] based on predictions about surrounding vehicles but this also adds a major load to the system. The algorithm presented here is able to deal with the raw readings and work in real time on a low capacity CPU.

Some researchers have already used velocity space optimization approaches for autonomous car-like local navigation. In [29] the Beam Curvature Method ([5]) was adapted to the local car-navigation problem by adding the roadside constraints. The motion planning method proposed for unstructured road environments combines the Lane Curvature method and the Beam Curvature method. Qijun and Xiuyan ([30]) proposed an improved BCM algorithm ([31]) for a security patrol robot used to inspect abnormal situations. The vehicle considers its kinematic and dynamic constraints. In the algorithm, the data acquired by the three layer sensors were mapped on a horizontal plane and then overlapping simplified beams were constructed from the two dimensional distribution of obstacles. However, all the velocity space optimization approaches mentioned so far assume a holonomic, cylinder-shaped vehicle or use the cylinder that includes the vehicle to define the collision area with obstacles. Although this is acceptable for a wide range of small differential and synchro mobile robots, it cannot be applied to car-like vehicles.

In this paper, we present a low-level reactive control (CVM-Car) that combines a pure pursuit strategy to calculate a reference and an extension of the CVM method that takes into account the shape and non-holonomic nature of a car. This approach allows the vehicle to stay centered in the lane and able to avoid unknown obstacles that can partially block the lane. The advantage with respect to previously mentioned approaches is that it includes the car-shape restrictions. Compared to the methods that use sampling-based trajectory planning, it is faster because it avoids the search and collision detection for each trajectory.

III. CVM-CAR OVERVIEW

Figure 1 shows the different components of the CVM-Car module. The inputs are the path (defined as a sequence of lanelets), provided by a high-level planner, and the sensor readings from the driver that controls a velodyne LIDAR sensor.

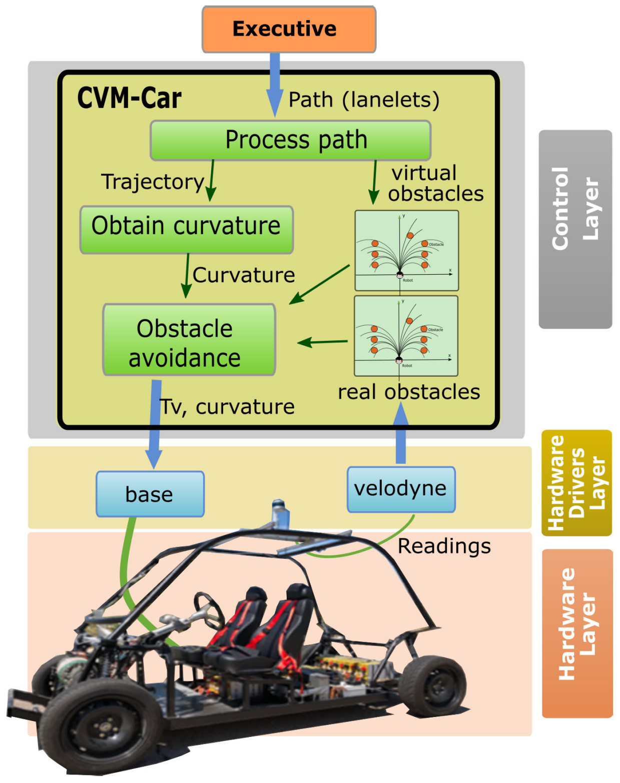


FIGURE 1. CVM-car structure and interface with the car.

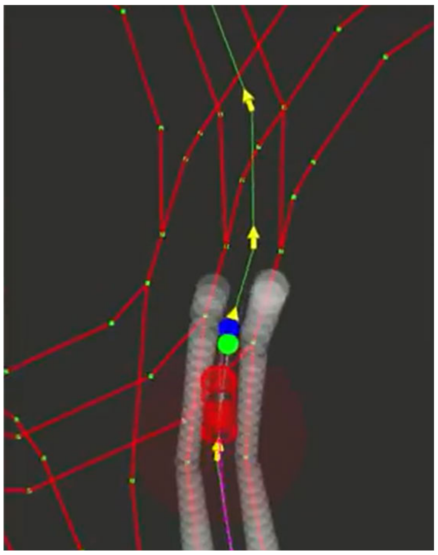
There are three main steps on the CVM-Car algorithm that correspond to the three green boxes in figure 1:

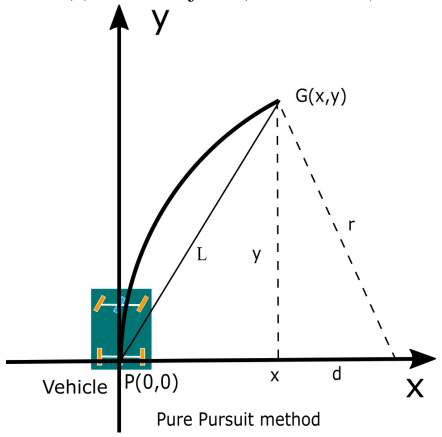
- **Process path.** Receives the path as a sequence of lanelets and produces the trajectory that the car should follow along with a set of virtual obstacles on the sides of the lane to keep the car inside the lane.
- **Obtaining the curvature.** This step uses the pure pursuit path-tracking algorithm to obtain the curvature reference to follow the path.

- **Obstacle avoidance.** Based on the curvature velocity methods, this step produces the velocity control commands to follow the curvature goal provided by the last step, while avoiding the obstacles detected by the sensorial system and the virtual obstacles generated by the *process path* step.

A. PROCESS PATH

The route comes from the high-level planner modules as a sequence of lanelets. Each lanelet is defined by the borders, called ways in [32]. Figure 2a shows one example of lanelets in a roundabout intersection (red lines) defined by the yellow dots and the car entering the roundabout. The planned path is the sequence of lanes with a green line in the middle. The green line is the trajectory obtained as a sequence of points in the center of the lane. The white circles on top of the lane borders are virtual obstacles. These virtual obstacles force the reactive module to keep the car in the lane.


(a) virtual objects (white circles)



(b) Curvature reference for car position in figure 2a

FIGURE 2. Car entering a roundabout. Obstacles, lanes, and curvature reference.

Virtual obstacles are obtained by simulating the reading of a laser range on a virtual wall located along the border lines. It consists of a list of points located on the border lines. We are currently working on a vision system to detect features of the road such as lane border lines to correct possible discrepancies with the map.

B. OBTAINING THE CURVATURE

This step is a basic implementation of the pure pursuit path tracking algorithm ([9]). This algorithm calculates a curvature that is used later by the obstacle avoidance step. The objective is to determine the curvature that will drive the vehicle to a chosen point on the path $G(x, y)$ that is one look-ahead distance L from the current vehicle position (figure 2b).

As shown in [9], the curvature c can be obtained from the equations:

$$r = \frac{L^2}{2x}; \quad c = \frac{1}{r} = \frac{2x}{L^2} \tag{1}$$

Implementation is also straightforward. The only parameter to select is the look-ahead value L. For short values, the vehicle will tend to oscillate and for long distances, the vehicle will cut the curves. In this case, the look-ahead distance is selected according to the speed of the car. Values range from 7 to 20 meters in a directly proportional way to the speed of the car, which is in turn inversely proportional to the curvature of the road. That is, the greater the curvature, the shorter the selected look-ahead. We will see later that the obstacle avoidance method will keep the vehicle from cutting the curves, even when relatively long look-aheads are selected.

C. OBSTACLE AVOIDANCE

The obstacle avoidance is based on similar principles developed by other curvature methods (such as CVM, BCM, and dynamic window) that are widely used in the field of autonomous mobile robots. However, some of the assumptions considered in most of these algorithms cannot be applied to a car-like vehicle. In a similar way to the curvature methods, CVM-Car formulates local obstacle avoidance as a constrained optimization problem in the velocity space of the vehicle. The algorithm uses the curvature calculated in the last section and the sensor readings as inputs (Figure 1). The curvature reference is the one the car should follow if there are no obstacles in that direction. The sensor readings are a list of distance points provided by different types of sensors, such as sonar or LIDAR. The outputs of this algorithm are the translational (*tv*) and rotational (*rv*) velocities that are sent to the vehicle to obtain the steering-wheel speed and throttle according to the Ackerman model.

The algorithm, like the curvature methods, assumes that the vehicle travels along arcs of constant curvature (c) that correspond to a point in the velocity space (*tv*, *rv*) during each cycle time. While this is an approximation, those effects can be negligible due to the effects of acceleration, especially for short cycle times (high frequency updating rates) where conditions do not change suddenly. The problem is then to find the optimal point in the area of the velocity space restricted by the dynamics of the vehicle (dynamic window) and the obstacles.

This algorithm includes two main steps that will be described in detail in the next sections:

- **Obtaining the constrained area in the velocity space**. This area is obtained from the dynamic restrictions of the vehicle and the restrictions imposed by the obstacles.
- **Obtaining the optimal point in the constrained area**. From the safe area obtained in the first step a point is selected that maximizes some objective function.

IV. OBTAINING THE CONSTRAINED AREA IN THE VELOCITY SPACE

As in CVM, we assume that the vehicle travels along arcs of constant curvature (c) that correspond to a point in the velocity space (rv, tv) during each cycle time. Therefore, the problem is to find the optimal point in the area of the velocity space restricted by the vehicle dynamics and the obstacles.

A. VEHICLE DYNAMIC RESTRICTIONS

The dynamics of the vehicle impose maximum and minimum rotational (rv) and translational (tv) velocities given the current velocities (rv_{curr}, tv_{curr}) and maximum accelerations (ra_{max}, ta_{max}):

$$\begin{aligned} rv_k - (ra_{max} \cdot T_{accel}) \leqslant rv_{k+1} \leqslant rv_k + (ra_{max} \cdot T_{accel});\\ 0 \leqslant tv_{k+1} \leqslant tv_k \leqslant tv_k + (ta_{max} \cdot T_{accel}) \end{aligned} \tag{2}$$

where T_{accel} is chosen according to the cycle time of the algorithm execution. These velocities are also limited by the kinematic restrictions as the maximum and minimal physical vehicle velocities:

$$\begin{aligned} -rv_{max} \leqslant rv_{k+1} \leqslant rv_{max};\\ tv_{k+1} \leqslant tv_{max} \end{aligned} \tag{3}$$

In addition to these limitations, the Ackerman configuration will impose limits on the vehicle' s steering. There is a maximum steering that will be associated to a maximum curvature $c_{maximum}$, which is usually the same steering to the right ($c_{maximum}$) and left ($-c_{maximum}$). This constraint is added to the kinematic and dynamic restrictions:

$$-c_{maximum} \leq \frac{rv_{k+1}}{tv_{k+1}} \leq c_{maximum} \tag{4}$$

Regarding the maximum linear velocity, CVM-Car adds some restrictions to those already imposed by CVM. The maximum linear velocity is the minimum of all the maximum velocities. There are two new restrictions on that value: the first one comes from the behavior decision layer [33] and the second one from the road curvature. The behavior decision layer sets a limit based on the traffic rules (traffic signs and limits depending on the type of road) and the specific situation (for example, if there is a pedestrian close to the car). Another limit of the linear velocity is determined by the curvature of the road. We use a look-ahead distance (35 m) to obtain the road curvature and then calculate the maximum safe speed. This is similar to what human drivers actually do: they slow down when entering a curve, and then speed up when leaving the curve.

The trajectory is a sequence of points $pi(x_i, y_i)$. The first step to calculate the curvature of the road is to obtain the angle between the two vectors defined by three consecutive points p_{i-1}, p_i and p_{i+1} (vector $\overrightarrow{p_{i-1}p_i}$ and vector $\overrightarrow{p_ip_{i+1}}$):

$$\alpha_i = \cos^{-1}\left(\frac{\overrightarrow{p_{i-1}p_i} \cdot \overrightarrow{p_ip_{i+1}}}{\left|\overrightarrow{p_{i-1}p_i}\right| \left|\overrightarrow{p_ip_{i+1}}\right|}\right) \tag{5}$$

In order to remove the noise produced by small errors in the trajectory when the trajectory points are located too close to each other, we use a moving average filter with a window of 4 meters. Finally, the parameter that imposes a restriction on the maximum speed due to the trajectory curvature is obtained:

$$\tau_i = \sum_{\left|\overrightarrow{p_ip_j}\right| < L} \alpha_j \tag{6}$$

where L is the look-ahead distance to take into account the curvature. In order to scale the parameter τ_i, a maximum angle value is set ($\tau_{MAX} = 100$ degrees) that corresponds to the minimum value for the maximum speed (MIN_V). The maximum velocity is inversely proportional to this parameter:

$$\begin{aligned} range &= MAX\_V - MIN\_V\\ \max tv_i &= MAX\_V - \frac{range}{MAX\_A} \min\{\tau_i, MAX\_A\} \end{aligned} \tag{7}$$

In our experiments, MAX_V was set to 50 km/h, which is the maximum speed on the campus area were tests took place, and MIN_V was set to 10 km/h. This is the maximum speed reference, meaning that the obstacle avoidance algorithm cannot drive faster but it might drive more slowly if needed (road with obstacles).

Figure 3 shows the trajectory followed by the car in a test carried out with our vehicle on the University of Alcalá campus (Madrid, Spain). Figure 3a shows the whole trajectory and figures 3b and 3c correspond to enlargements of the roundabouts sections. Figure 4 illustrates the curvature correction parameter τ and the maximum speed according to the road curvature.

The curvature correction parameter has very high values when entering a curve and almost zero when the vehicle faces a straight lane. When the car starts, it faces a 90-degree curve to the right so the correction parameter is quite high. As the vehicle leaves the curve, there is a short straight lane and the correction parameter drops but soon reaches a high value again before the 90-degree intersection. Once on the main road, there is a new straight lane and the vehicle can reach maximum speed until it nears the roundabout. It can be seen that about 30 meters before the roundabout, the maximum

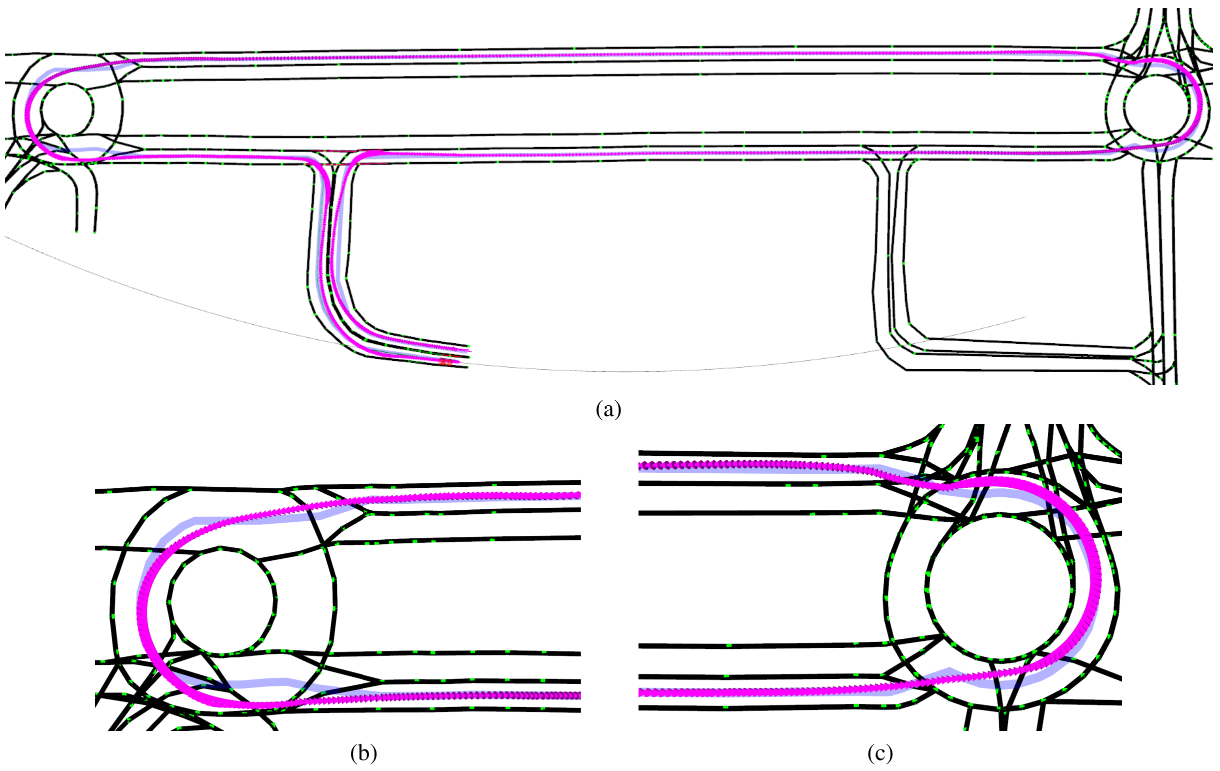


FIGURE 3. Car starts on the road at the bottom of figure, goes to the roundabout on the right, then to the roundabout on the left and returns to the starting point.

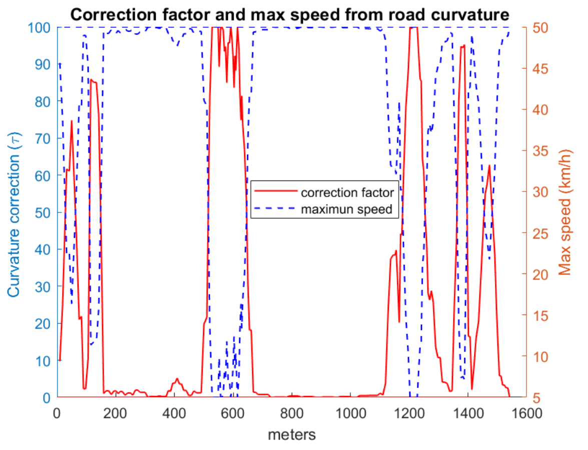


FIGURE 4. Maximum speed correction parameter and maximum speed for the trajectory in figure 3. The maximum speed decreases about 30 m. before a curve and increases inmediatelly after the car finishes the curve.

speed is reduced (high τ) and on leaving the roundabout the maximum speed is increased again (low τ).

B. OBSTACLE RESTRICTIONS

The obstacle avoidance method has to deal with obstacles of different shapes. If the obstacles are included on the map, they are modeled by a sequence of circles on the perimeter of the obstacle. Obstacles detected by the sensors such as a laser scanner are modeled as circles centered on the points obtained by the scanner.

Most of the curvature methods assume the vehicle is circle-shaped. That assumption avoids the high CPU-consuming convolution process to check if the vehicle describing a curvature will collide with an obstacle. In our case, the car shape cannot be considered as a circle because the use of a circumference that includes the car would be too restrictive and this could cause the car not to pass through narrow openings. On the other hand, the use of the car width as the diameter of the circle will cause a collision between the front of the car and some obstacles when turning. In this section, the Ackerman configuration and the rectangular shape of the vehicle are taken into account to adapt the obstacle avoidance method.

The constraints due to obstacles are also restricted areas in the velocity space, but now the car shape has to be taken into account to obtain the collision intervals. The main steps of the algorithm to add obstacle restrictions are summarized in figure 5. Each circle defines a curvature interval (c_{min}, c_{max}) that is characterized by the distance to impact d (figure 5a). Transforming this area from the x-y space to the velocity space (translational tv vs rotational rv) results in an area delimited by three straight lines (figure 5b). Following these steps for all the obstacles, the non-collision area in the velocity space is delimited to the intersection of the curvature intervals and dynamic limitations rectangle (figure 5c).

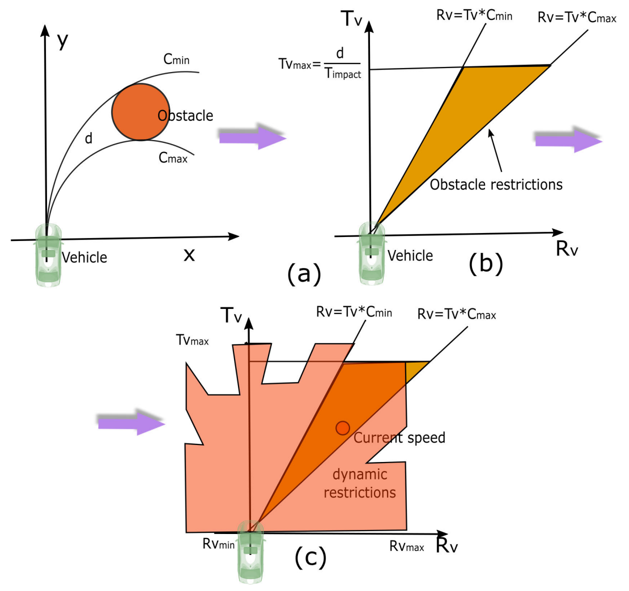


FIGURE 5. **Restrictions imposed by the obstacles.**

The following subsections describe the way to obtain the curvature interval and the collision distance.

1) CURVATURE INTERVAL

The curvature interval in CVM is defined by the maximum curvature, minimum curvature and impact distance. The CVM algorithm considers the vehicle to be circular. Therefore, converting Cartesian to configuration space obstacles merely involves increasing the radii of the obstacles by the radii of the vehicle. The curvature interval associated to an obstacle (obs) are the curvatures between c_{min} and c_{max}.

The changes needed on the curvature interval to take into account the vehicle' s shape are analyzed in this section. Figure 6a shows that the minimum curvature for the collision interval with the obstacle located in (x_{obs}, y_{obs}) is the same as if the vehicle were a circle with diameter the width of the car (B). The vehicle is going around the obstacle and the front part is always further away from the obstacle. Therefore, the minimum curvature can be calculated according to [4]:

$$c_{min} = \frac{2\,(r_{obs} + x_{obs})}{x_{obs}^2 + y_{obs}^2 + r_o^2} \tag{8}$$

The worst-case scenario for the maximum curvature is depicted in figure 6b. The front of the vehicle sticks out from the round shape calculated by CVM, and therefore the curvature interval is wider because the maximum curvature increases. Figure 6b shows that the part sticking out from the circle shape is Δr. This situation is therefore the same as increasing the obstacle radius by Δr.

From the triangle with vertices T_{11}, T_{12} and T_{13}, using Pythagoras' theorem:

$$(L - d_1)^2 + (r + r_{obs})^2 = (r + r_{obs} + \Delta r)^2 \tag{9}$$

Likewise, applying Pythagoras' theorem in the triangle with vertices P_o, T_{22} and T_{13}:

$$y_{obs}^2 + (r - x_{obs})^2 = (r + r_{obs} + \Delta r)^2 \tag{10}$$

Finally, from both equations, the maximum curvature can be obtained as:

$$c_{max} = \frac{2\,(r_o + x_{obs})}{x_{\text{obs}}^2 + y_{\text{obs}}^2 - r_o^2 - (L - d_1)^2} \tag{11}$$

As figure 6b shows, the maximum curvature has been increased with respect to the original CVM maximum curvature to take the shape of the vehicle into account.

2) COLLISION DISTANCE

The collision distance is the distance $d_c(c, obs)$ that the point vehicle would travel before hitting the first obstacle obs. As in [4], the collision distance can be obtained from (figure 6d):

$$d_c(c, obs) = \begin{cases} y_i & c = 0 \\ \left|\frac{1}{c}\right|(\theta) & c \neq 0 \end{cases} \tag{12}$$

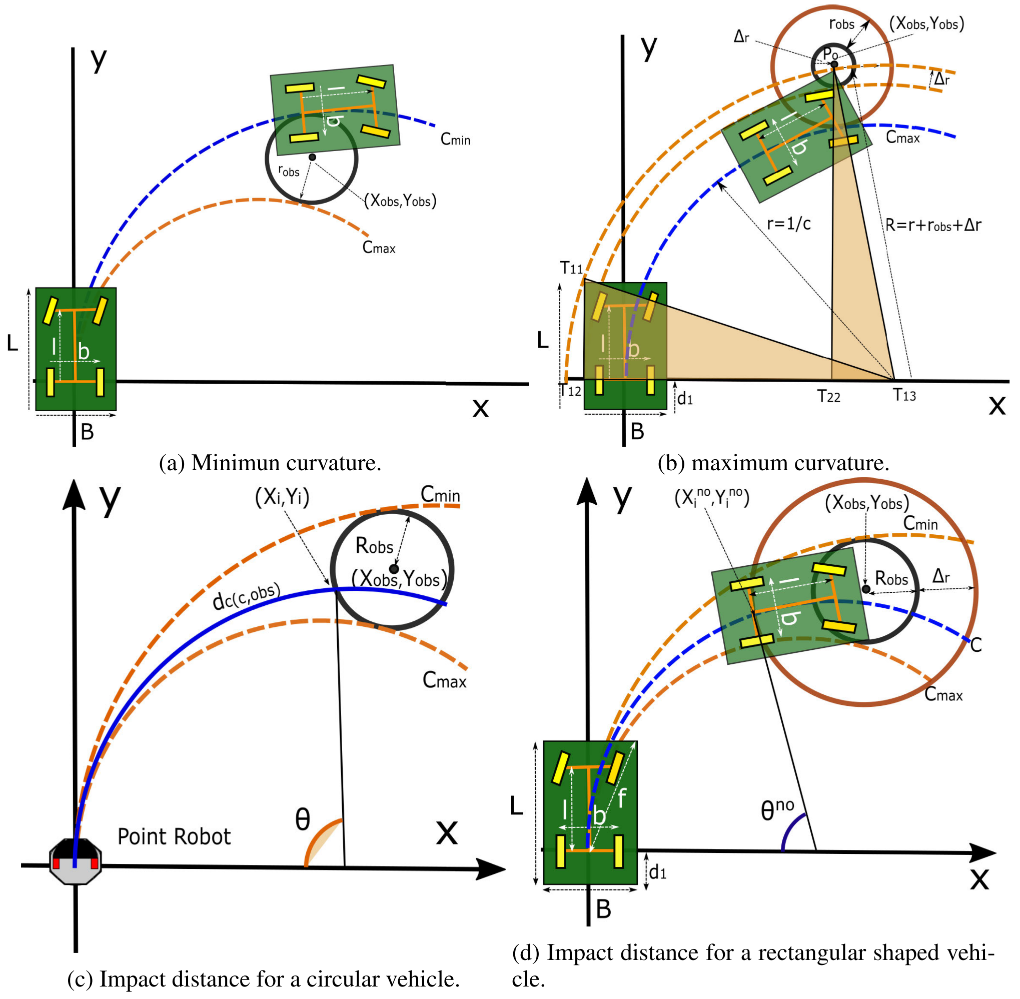
where (x_i, y_i) is the point at which the vehicle traveling along the curvature c intersects the obstacle and θ is calculated using the following equation:

$$\theta = \begin{cases} \text{atan}\left(\frac{y_i}{\left(x_i - \frac{1}{c}\right)}\right) & c < 0 \\ \pi - \text{atan}\left(\frac{y_i}{\left(x_i - \frac{1}{c}\right)}\right) & c > 0 \end{cases} \tag{13}$$

A similar situation is depicted in figure 6d, but in this case with the vehicle shape represented as a rectangle. The collision distance changes because the front of the vehicle sticks out from the circle shape calculated by CVM. The collision distance in CVM is discretized and used in the evaluation function to indicate a preference for traveling longer distances along the given curvature without hitting obstacles. Hence, the use of a conservative assumption will not entail the possibility of missing a free opening as could happen with the computation of curvature intervals. Therefore, to obtain the collision distance we enlarge the obstacle radii by the diagonal *f* (worst-case scenario) instead of by *B/2*, as shown in figure 6d. By using this approach, we also make sure that the selected speed will avoid a potential collision.

The collision distance is calculated using the following equation:

$$d_c(c, obs) = \begin{cases} y_i - \left(\frac{L}{2} - \frac{B}{2}\right) & c = 0 \\ \left|\frac{1}{c}\right|(\theta^{no}) & c \neq 0 \end{cases} \tag{14}$$



(a) Minimun curvature.

(b) maximum curvature.

(c) Impact distance for a circular vehicle.

(d) Impact distance for a rectangular shaped vehicle.

FIGURE 6. Parameters that characterize a curvature interval.

where θ^{no} is obtained using (x_i^{no}, y_i^{no}) instead of (x_i, y_i):

$$\theta^{no} = \begin{cases} atan\left(\frac{y_i^{no}}{\left(x_i^{no} - \frac{1}{c}\right)}\right) & c < 0 \\ \pi - atan\left(\frac{y_i^{no}}{\left(x_i^{no} - \frac{1}{c}\right)}\right) & c > 0 \end{cases} \quad (15)$$

The point (x_i^{no}, y_i^{no}) is the intersection of the curvature with the obstacle with radii increased by Δr:

$$\Delta_r = \sqrt{\left((L - d_i)^2 + \left(\frac{B}{2}\right)^2\right)} - \frac{B}{2} \quad (16)$$

V. OBTAINING THE OPTIMAL POINT IN THE CONSTRAINED AREA

Once the area with permitted velocities is defined, the next step is to choose a point in linear-angular velocity space (tv, rv), which satisfies the constraints already defined and maximizes an objective function $f(tv, rv)$. This objective function attempts to move the vehicle close to the reference curvature (path) at the highest feasible speed, while traveling with a longer collision-free space in its trajectory:

$$f(tv, rv) = \alpha_1 speed(tv) + \alpha_2 dist(tv, rv) + \alpha_3 head(rv) \quad (17)$$

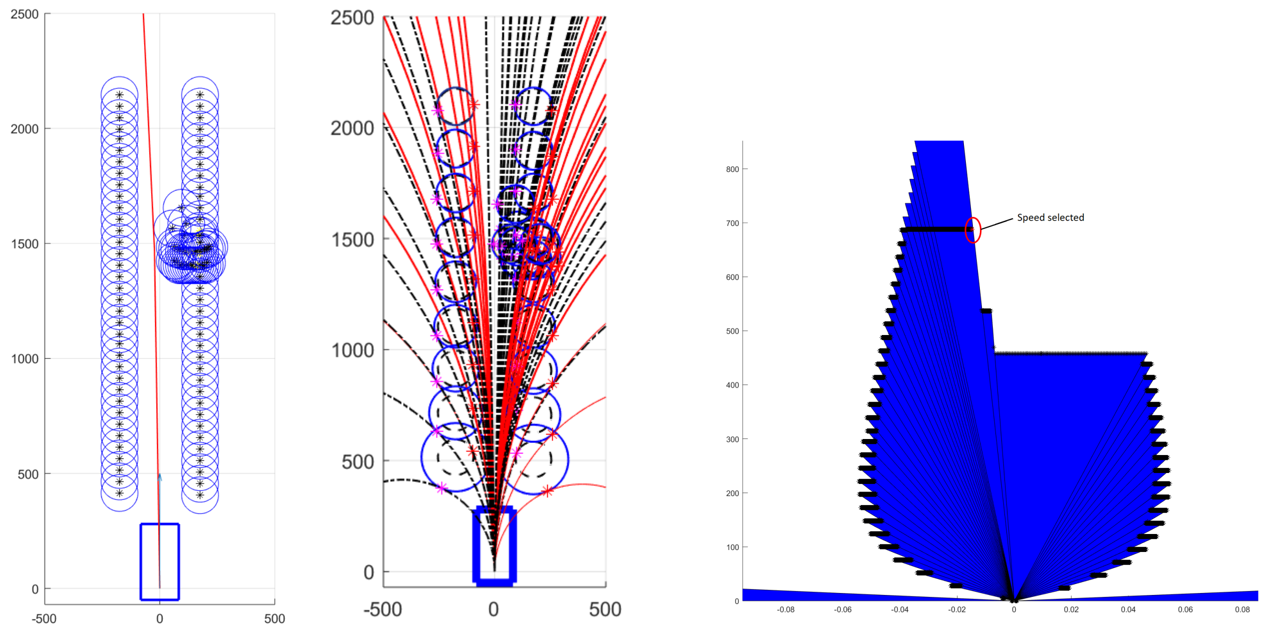
The α_i values indicate the relative weight to be given to each i-term in the objective function. The *speed* term indicates a preference for traveling faster:

$$speed(tv) = \frac{tv}{tv_{max}} \quad (18)$$

The *dist* term indicates a preference for traveling longer distances along the given curvature without hitting obstacles:

$$dist(tv, rv) = \frac{D_{limit}(tv, rv, OBS)}{L} \quad (19)$$

The *head* term is the goal heading error. The goal reference in the curvature velocity methods is a direction. However, in the CVM-Car algorithm the reference is already a curvature. This is therefore a more direct method: the goal is a



(a) Obstacles. (b) Interval curvatures (space). (c) Interval curvatures (velocity space)

FIGURE 7. **Interval curvatures created from the virtual obstacles on the lane border and a real obstacle on the right side of the lane.**

curvature obtained by the pure pursuit step; the curvature to be evaluated can be compared with the goal curvature; and therefore the *head* term can be computed as:

$$head(tv, rv) = \frac{0.001}{max\left(\left|cg - \frac{rv}{tv}\right|, 0.001\right)} \tag{20}$$

where c_g is the reference curvature provided by the pure pursuit method.

Figure 7 shows a straight-lane scenario with an obstacle partially blocking the lane ahead on the right. Figure 7a shows the virtual obstacles on the lane borders and the real obstacles added from the sensor readings. Figure 7b shows the curvature intervals in Cartesian space. On the other hand, figure 7c shows the corresponding intervals in the velocity space. The linear velocity (axis y) on each interval is limited by the impact distance, with the exception of the front left interval, which is limited by the maximum speed of the car at that point. The selected velocity point that maximizes the objective function is the red dot inside the red circle.

A. DRIVING CLOSE TO OBSTACLES

Some curvature methods, such as CVM, include a constraint that makes the maximum allowable translational velocity proportional to the distance between the selected curvature and closest obstacle ([4]). That is, if the curvature selected for the car to travel passes close to an obstacle, then the translational velocity is limited. In other words, the speed must be reduced when navigating close to an obstacle.

This restriction cannot be directly imposed in the CVM-Car method. There are situations where the car has to travel quite fast close to obstacles such as the road guardrails. However, it is still better to drive as far from the obstacles as possible. Therefore, instead of using the distance between obstacles and curvature to define a maximum allowable translational velocity, we use the distance to the obstacles as another weight term in the evaluation function to express the preference to drive far from the obstacles if possible.

This term is only computed for the points belonging to curvatures that are outside of the obstacle intervals because the curvatures inside the obstacle intervals hit the obstacle and the distance is zero.

Consequently, the fitness function has a new term:

$$\begin{aligned} f(tv, rv) = \alpha_1 speed(tv) + \alpha_2 dist(tv, rv) \\ + \alpha_3 head(rv) + \alpha_4 curvDist(tv, rv) \end{aligned} \tag{21}$$

where the new term *curvDist* is the minimum distance to the obstacles of the curvature arc described by the car if traveling with velocity (tv, rv).

For a velocity point (tv, rv), the distance to an obstacle located at (x_{obs}, y_{obs}) with radius r_{obs} is (Figure 8a):

$$d_{obs} = max\{|\sqrt{\left(\left(x_{obs} - \frac{1}{c}\right)^2 + y_{obs}^2\right)} - \left|\frac{1}{c}\right|| - r_{obs}, 0\} \tag{22}$$

where c is the curvature ($c = rv/tv$) and the inverse $(1/c)$ is the curvature radius. Finally, the *curvDist* term is obtained as:

$$curvDist(tv, rv) = \frac{\min_{obs \in OD_c}\{d_{obs}, MAX\_D\}}{MAX\_D} \tag{23}$$

where OD_c for curvature c with impact distance d_{obs_c} is defined as the set of obstacles that have an impact distance

d_{obs_i} at least MIN_D shorter than d_{obs_c}:

$$OD_c = \{obs_i / d_{obs_i} < (d_{obs_c} - MIN\_D)\} \tag{24}$$

Adding this new weight keeps the vehicle in the center of the lane.

To find the point with maximum value for the objective function in the restricted area, we proceed as in CVM. First, it should be noted that each curvature interval provides a pair of linear inequalities on the velocity space. The dynamic constraints are also linear inequalities and the objective function is also linear and monotonically increasing in tv.[1] Furthermore, since the distance value is constant between pairs of curvature lines, the optimal value of the function, for each curvature interval, lies along the top boundary of the constraint lines. Thus, we can proceed to obtain the maximum in a very efficient way by choosing the overall best value of the objective function for these points:

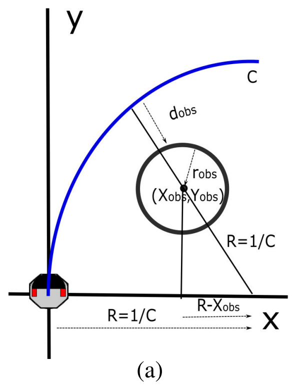
- For each curvature interval, each vertex along the upper constraint boundary.
- For the goal curvature, the point that intersects the upper constraint boundary.

For wide curvature intervals we also included some samples along the upper constraint boundary in order to obtain a better approximation to the maximum of the last term of the fitness function.

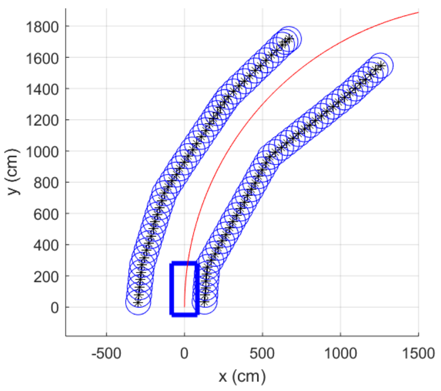
B. ACCOUNTING FOR THE OBSERVATION-ACTION DELAY

The laser is providing distances to obstacles every 100 msec. There is a delay between the time those distances are measured by the laser and the time reactions to those distances are effective in the car. Experimental measurements in our system show that this delay is about 220 msec. This situation is similar to that experienced by humans. Some studies [34] show that human reaction times, measured as the sum of perception time and the time of leg transfer from the accelerator pedal to the brake pedal, range between 0.5 and 1.2 seconds. For the autonomous car, this means that the reactive control is making decisions about distances that have been taken in the past (sensing time delay "st") and the actions will start taking effect in the future (action time delay "at"). The distances were taken when the car was in position $P_t(x_t, y_t, \theta_t)$ and the action is taken later when the car is in position $P_{t+\Delta t}(x_{t+\Delta t}, y_{t+\Delta t}, \theta_{t+\Delta t})$. However, the ideal situation would be to make the decision about the state when the action is carried out. The way we approach this is by first predicting the car position $P_{t+\Delta t}(x_{t+\Delta t}, y_{t+\Delta t}, \theta_{t+\Delta t})$ after the total delay ($\Delta t = at + st$), then obtaining the readings transformation to the new coordinate system and finally applying the algorithm described here.

[1] We are aware that the last term of the fitness function does not hold this. However, the benefit obtained by the efficiency of the algorithm justifies this approach.



(a)



(b)

FIGURE 8. a) Distance between arc curvature C and obstacle. b) Virtual and real obstacles (blue) entering a curve and selected curvature (red).

To predict the increment of the car position we use a simple vehicle model considering that Δt is a small period of time:

$$\begin{aligned}\Delta\theta &= \frac{v_t}{L_f}\Delta t tan(\delta_t) \approx \frac{v_t}{L_f}\Delta t\delta_t;\\ \Delta x &= \frac{v_t}{\Delta\theta}\Delta t 2\left(sin(\frac{\Delta\theta}{2})\right)^2 \approx v_t\Delta t\frac{\Delta\theta}{2};\\ \Delta y &= \frac{v_t}{\Delta\theta}\Delta t 2 sin(\frac{\Delta\theta}{2})cos(\frac{\Delta\theta}{2}) \approx v_t\Delta t\end{aligned} \tag{25}$$

where δ_t is the steering angle and L_f is the distance between the car axes. The reading coordinates (x_i, y_i) are referred to the car local coordinate system. That is, the car position P_t when the readings were taken. To convert the original reading coordinates (x_i, y_i) to the new ones (x'_i, y'_i) referring to the car's new position $P_{t+\Delta t}$, we use the following transformation:

$$\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} cos(\Delta\theta) & -sin(\Delta\theta) \\ sin(\Delta\theta) & cos(\Delta\theta) \end{pmatrix} * \begin{pmatrix} x_i - \Delta x \\ y_i - \Delta y \end{pmatrix} \tag{26}$$

VI. RESULTS AND CONCLUSION

The local navigation method described here has been implemented in a ROS node according to the architecture presented in [10]. The inputs and outputs of this node (figure 1) correspond to ROS messages with other modules of the software architecture. The system has been tested first in a V-REP simulator ([11]), then in the open-source car simulator CARLA ([12]) and also in an autonomous vehicle prototype.

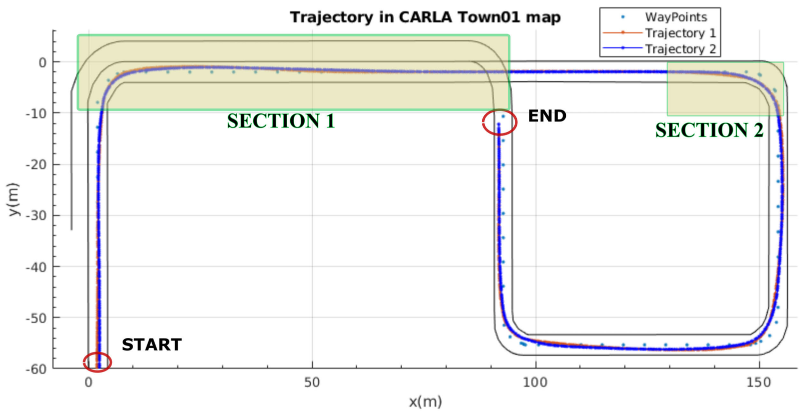


FIGURE 9. Car trajectory on town 1 CARLA map.

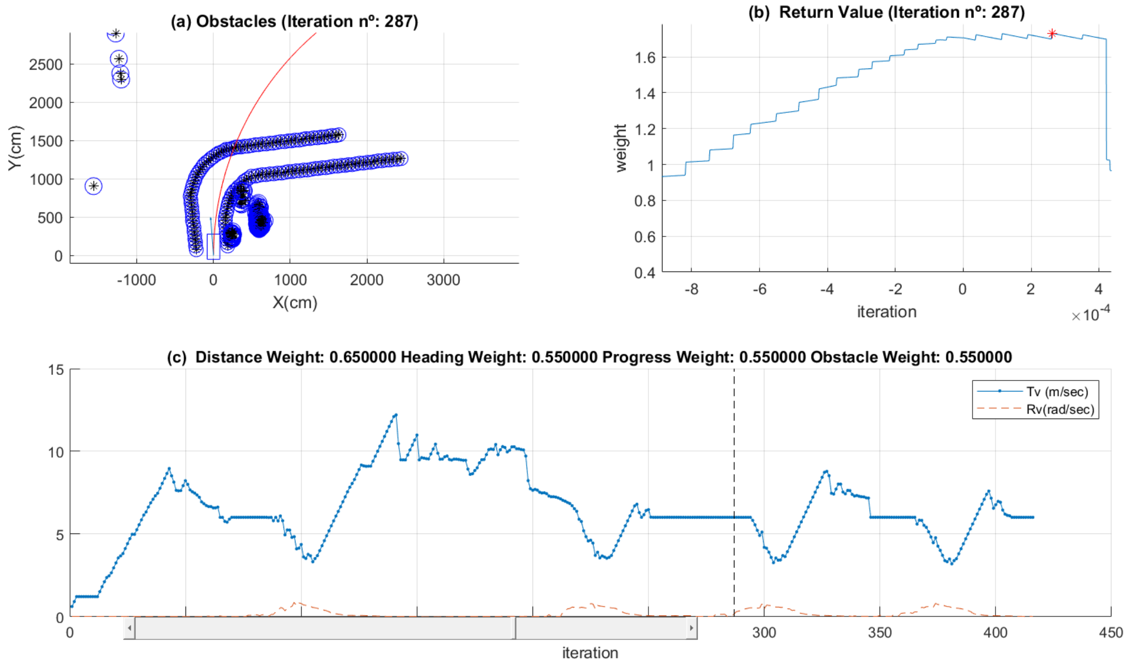


FIGURE 10. Figure(c) shows the translation and rotation speeds while the car follows the trajectory in figure 9. Figure (a) shows the obstacles (blue circles) and best curvature (red) when the car is in a curved section of the path and figure (b) shows the results of the evaluation function at the same instant.

A. CARLA SIMULATION TESTS

CARLA is an open-source simulator for autonomous driving research with different urban environments and environmental conditions. First, a simple simulation scenario will illustrate the role of the different steps described in the previous sections. The scenario is one of the urban environments provided with CARLA (Town 1) with a total of 2.9 km of drivable roads.

Figure 9 shows the lanes (black), the waypoints (points in blue) and two trajectories (dark blue and red) from two different executions. The difference between the two trajectories that overlap most of the way will be discussed later.

Trajectories followed by the car are very smooth and very similar to the ones driven by a human. Waypoints are always in the center of the lane but, unlike previous approaches, the vehicle is not strictly following the waypoints. Instead, a perception system defines the borders of the lane and CVM-Car keeps the car inside the lane; similar to what a human driver would do. For example, when taking a curve, human drivers tend to approach the inner side of a curve and, when leaving the curve, tend to get closer to the outer side. That is the behavior observed in figure 9.

A close analysis of the different fitness function weights when the vehicle is entering a curve will illustrate the decisions taken by CVM-Car. Figures 10 and 11 represent a

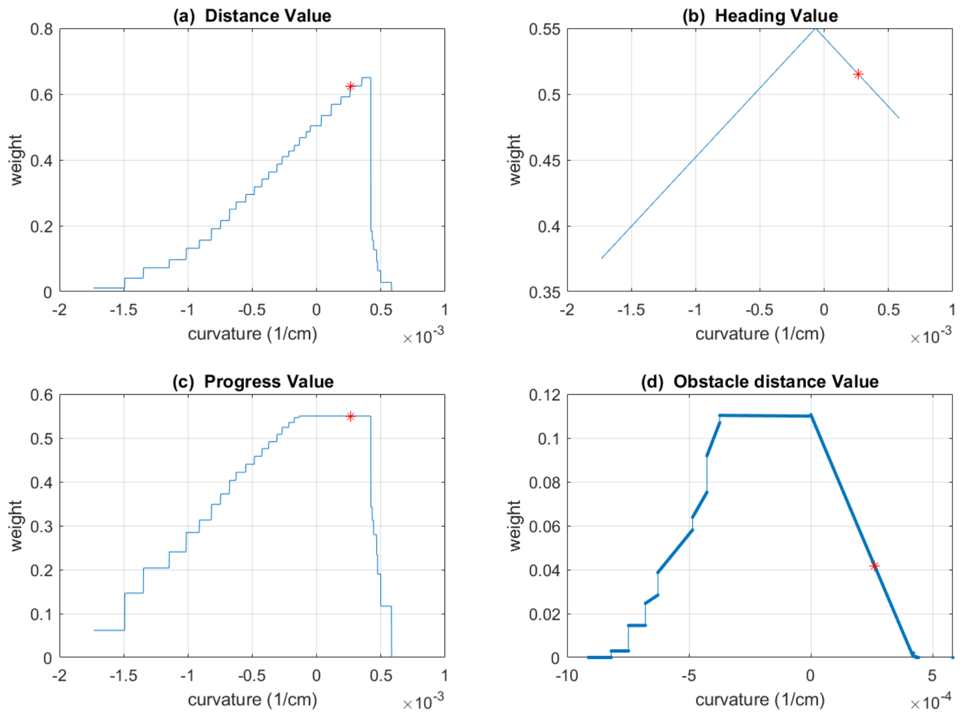


FIGURE 11. Four components (weights) of the evaluation function for the same time instant represented in figure 10a.

snapshot of different parameters when the vehicle is driving in a curved section of the road. Figure 10c represents the linear and angular velocities of the car for the trajectory in figure 9. The four valleys correspond to the four curves in the trajectory. The vertical dashed line represents the point selected for the snapshot (iteration 287). Figure 10a shows the obstacles (blue circles) and the curvature selected (red line). Finally, figure 10b shows the value of the fitness function for the range of curvatures evaluated. Curvatures close to the curvature of the curve correspond to higher fitness value.

As discussed in section V, the fitness function includes four different components. The weight of each component on the final value for each curvature is shown in figure 11. The distance component for this situation has the maximum values for curvatures closer to the inner part of the curve since they correspond to curvatures where the car can travel a longer distance before hitting an obstacle (highest impact distance). On the other hand, the obstacle distance weight tries to keep the car from driving close to obstacles. The highest fitness value (red dot) is a curvature that passes close to the inner part of the curve but not too close. The heading direction is defined by the next waypoint that, in this case, is the one with curvature zero or close to zero. Therefore, the heading weight decreases linearly as the curvature moves away from this value according to section V. The progress value is the ''projection'' of the distance for the next interval in the goal direction.

The distance to obstacles is the component of the fitness function that favors trajectories away from obstacles.

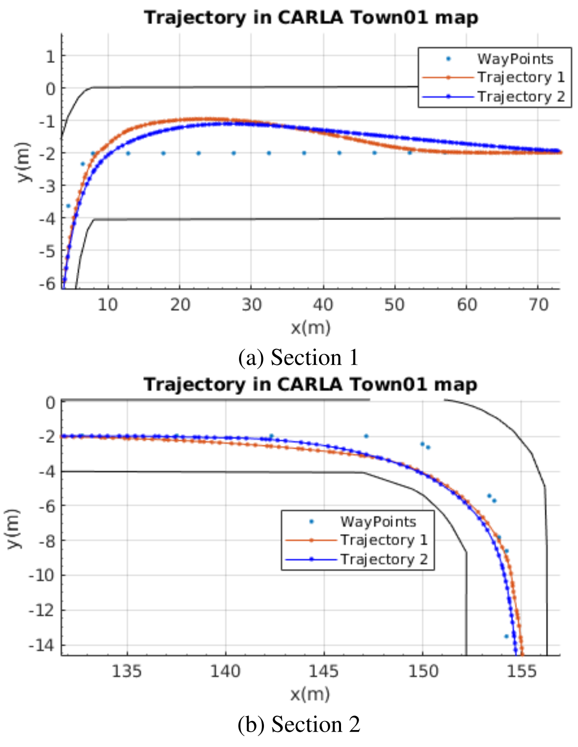


FIGURE 12. Car trajectories for two sections on town 1 CARLA map (figure 9).

To illustrate the effect of the distance to obstacles weight, figures 12a and 12b show in detail two sections of the trajectory for two different executions (red and blue lines).

In the first execution (trajectory 1 in red) the fitness function does not include the distance to obstacles weight while in the second one (trajectory 2 in dark blue) it is included. For the straight parts of the trajectory when the car is in the middle of the lane there is not much difference because the direction and progression weight keep the car in the middle. However, while the car is heading towards the curve, the first trajectory (red) heads to the inner part of the curve sooner. This could be a problem because at the end it means a sharper turn in the curve. When the vehicle is finishing the curve, a similar effect is observed. The trajectories described with the weight that discourages trajectories too close to obstacles (blue) are smoother.

The speed limit in the area was close to 14m/sec, which is the typical speed limit for an urban area. Figure 13 shows the speed limits imposed by different factors in the first part of the trajectory. Because of the relatively low speeds, the less restrictive limit is the one imposed by the lateral acceleration in order to avoid the vehicle sliding laterally or tipping over on sharp turns. The impact distance is the more restrictive component in the curves because it is the one that makes sure that the vehicle, moving at the commanded curvature and speed, is able to stop within the impact distance. The impact distance labeled as "limit by CVM" includes the dynamic restrictions mentioned in section IV-A.

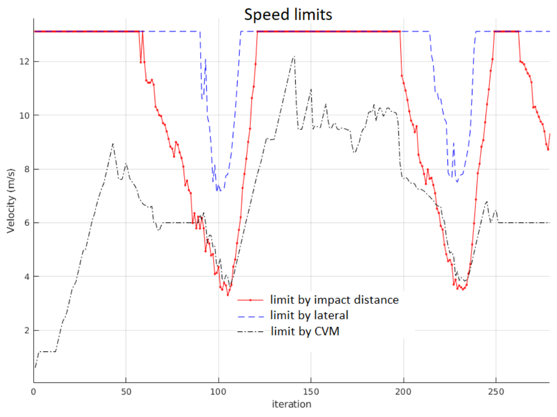


FIGURE 13. Speed limits for the first part of trajectory in figure 9 on town 1 CARLA map.

A more complex test scenario with other cars, pedestrians and bicycles is shown in the attached video. It shows that the car is forced to stop when an obstacle blocks the lane. However, the car is able to avoid the obstacle if the lane is only partially blocked.

B. ROAD TESTS

The car shown in figure 1 is based on an electric TABBY EVO from the Open Motors Company (Tabby Evo, 2018). The automation of the car was carried out by the University of Alcalá ROBOSAFE team. The local navigation module commands the translational velocity and the rotation angle of the front wheels. It uses the odometry and the readings from a LIDAR Velodyne (VLP-16) located on top of the vehicle.

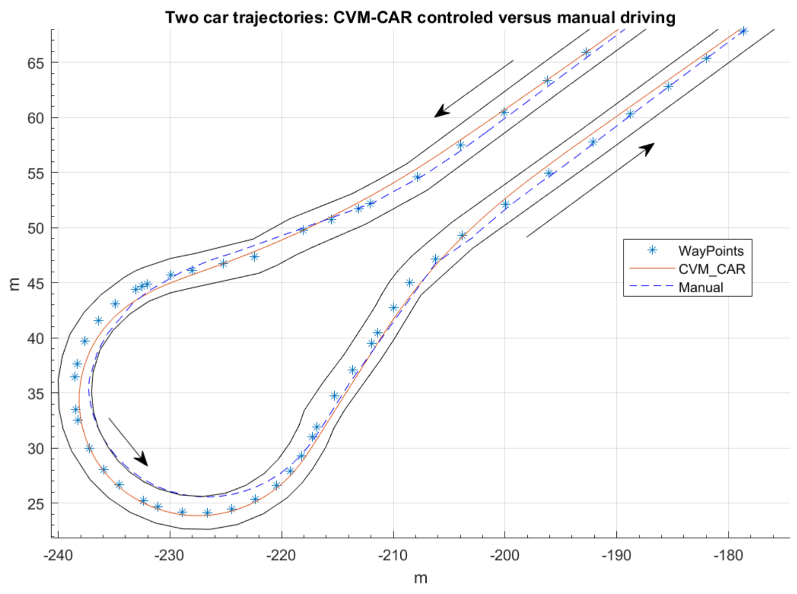
Several tests have been carried out with the TABBY EVO car in two environments: the campus of the University of Alcalá de Henares and the closed circuit used in the IROS 2018 autonomous vehicle demonstration event, both in Madrid.

Figure 3 shows the route followed by the car when traveling between two roundabouts on the University of Alcala de Henares campus. There are two lanes on each way and the car has to switch lanes in one of the roundabouts. Figure 3a shows the entire path and the two images at the bottom show a zoomed view of the trajectory in the roundabouts. Since the approach here is to drive within the lane instead of following a specific trajectory, we can see that the behavior is quite different from other approaches. The car uses the full width of the lane. According to the fitness function, the car tends to drive in the longest non-collision arcs, which is why it drives in the inner part of the curves. On the straight part of the route, the car tends to stay in the middle of the lane because of the preference to drive away from the "virtual" obstacles on the sides of the lane. Another advantage of this method is that by changing the weights of the fitness function we can change the driving behavior preferences.

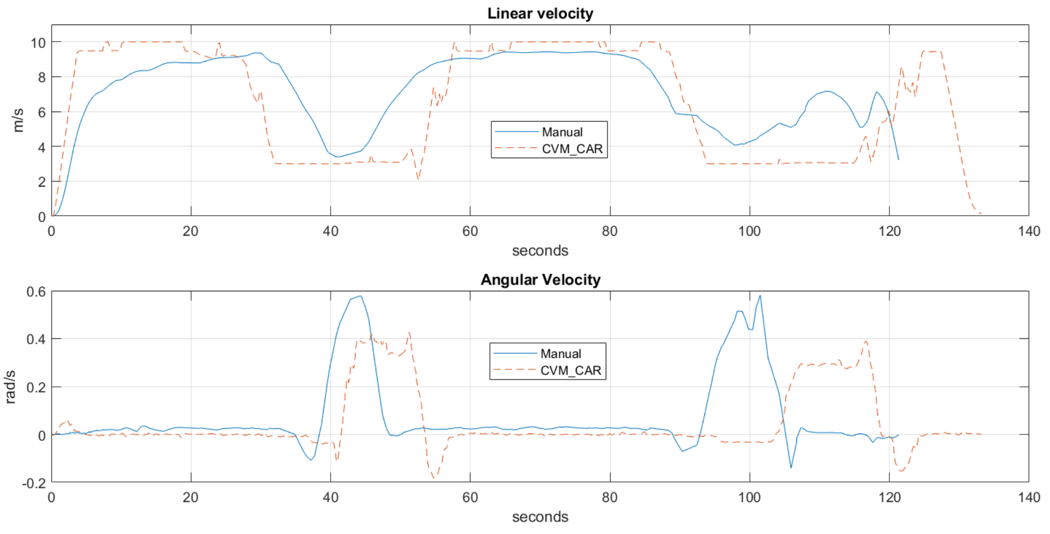
We have also compared the CVM-Car control with a human driving along the path in figure 3. The goal here is not to carry out extensive research on human driving since different people will take different decisions but to get an idea of the different decisions that a human driver takes compared to CVM-Car. For that purpose, we have chosen one of our students. One person is in charge of making most of the tests as he has more experience driving the car.

For the straight sections of the path both trajectories are very similar but CVM-Car always keeps the car in the center of the lane while the human driver allows for small deviations. Figure 14a focuses on one of the two roundabouts to appreciate the difference in both trajectories. As in simulation, the final trajectory with CVM-Car is very smooth, even smoother than the one described with the human driver. However, as the car enters the roundabout, CVM-Car approaches the inner part of the curve too soon (right side of the lane). However, the human driver does the opposite, first moving to the outer side (left side of the lane) and then, when already in the curve, moving to the inner part. Another interesting issue that can be observed in figure 14a is that after the first section of the roundabout, the CVM-Car trajectory is located in the center of the lane instead of the inner part because of the preference to drive in curvatures with the longest impact distance. Finally, when leaving the roundabout, CVM-Car approaches closer to the left side of the lane than the human driver. The main reason for this behavior is again the preference for curvatures with the longest impact distance. As a matter of fact, in this case this is the best technique for driving on a curve.

In order to assess the ride comfort evaluation, we compare the linear and angular velocity variation of our method



(a) Trajectories described by the car for two different driving systems



(b) Car velocities for two different driving systems

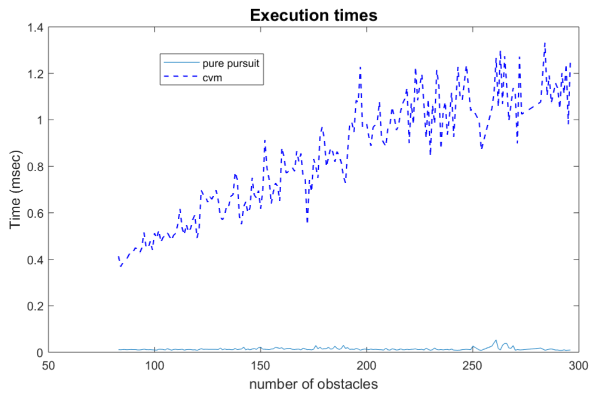
FIGURE 14. Comparing two different driving systems: CVM-Car and manual driving.

with the ride of a human driver (ISO 2631-4). This comfort evaluation method was also used by other researchers ([29]). The velocities registered by the car for the trajectories in figure 14a are shown in figure 14b. The maximum linear speed is reached at the straight sections of the road and it is slowed down when turning in the roundabouts. The human driver finished the path about ten seconds before the CVM-car basically because the person reduces the speed closer to the curves than the autonomous case. Regarding the linear velocity, even though there is not much difference between the two cases, the changes in linear speed for the CVM-Car case are a little bit sharper. Comfort was not an issue taken into account in the design of CVM-Car. The method obtains the best curvature and then sets the linear velocity as the maximum possible according to several restrictions. We did not consider passenger comfort among those restrictions because it did not seem to be an issue in the tests carried out. However, this parameter can easily be added in the future because the method to obtain the curvature and the linear velocity limits will remain the same.

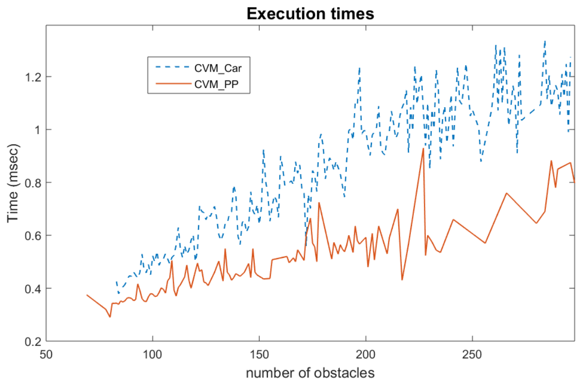
The angular velocity is close to zero in the straight sections of the road. However, a peak in angular speed in the first seconds shows that CVM-Car corrects faster a small deviation from the center of the road faster and then remains with angular speed closer to zero than the human driver.

1) EFFICIENCY ANALYSIS

In order to evaluate the efficiency of the approach presented in this paper, real-time data were recorded for both steps of



(a) Execution time of the *obtain curvature* step and the *obstacle avoidance* step.



(b) Overhead introduced in the execution times.

FIGURE 15. Execution times and overhead while the vehicle navigates the route described in figure 3.

the algorithm (pure pursuit and obstacle avoidance) while the vehicle navigates the road between the two roundabouts shown in figure 3. The average times for these executions are displayed in figure 15a. It was executed on an Intel(R) Core(TM) i7-6700 CPU at 3.40GHz.

The times obtained are grouped according to the number of obstacles detected. The time presented is the average time of all the cycles that manage the same number of obstacles. Figure 15a shows that the time for the *obtain curvature* step is independent of the obstacles, as expected, and is always less than 0.03 msec. Execution times for the *obstacle avoidance* step increase with the number of obstacles but it is always below 1.4 msec.

The overhead execution time produced by all the changes to adapt the method for a car-like vehicle is shown in figure 15b. CVM_PP is a method that uses pure pursuit to get the curvature reference and the original CVM. The overhead is less than 0.4 milliseconds.

C. CONCLUSION

The navigation algorithm proposed in this paper is safe because it avoids obstacles that do not block the lane. Furthermore, the algorithm keeps the vehicle within the lane. In addition, from the experiments presented here, local navigation is very rapid because it is based on simple, fast algorithms (pure pursuit and CVM). The obstacle avoidance step, unlike other CVM based methods, takes the kinematic restrictions and shape of a car-like vehicle into account. The final trajectory is very smooth with no sharp turns and with very low angular speeds.

It is important to notice that obstacles can have any shape. They will be represented as a sequence of points in their perimeter that usually correspond to sensor measurement points. Those points are converted into overlapped circles in the obstacle avoidance step of our method. That is the reason why this method can deal directly with raw data from 2D sensors or 3D sensors mapped in a 2D occupancy space.

The approaches in the literature that solve the navigation problem as a search problem ([3]), not only consider the next action but also the sequence of actions in the near future to follow the motion goal provided by the behavior decision module. This forward planning can include complex maneuvers and can obtain better solutions to decisions than the one presented here. However, due to the continuous nature of the decisions in each step (velocities in most cases), it is impossible to take the unlimited number of solutions into account and that is why some kind of discretization is needed. Therefore, for these search-based solutions, a possible sequence of short-term actions will produce a candidate trajectory. There are different ways to generate the candidate trajectories such as using a model-predictive trajectory generator ([14]). All the candidate trajectories need to be evaluated according to some parameters to select one of them. There is a tradeoff between execution time and resolution on the candidate trajectories. Using a coarse resolution will speed up the process, but increases the chances of missing narrow openings for example. Instead, CVM-Car is oriented to obtain a fast reactive control solution.

On the other hand, it should be pointed out that the CVM-Car method is not aimed at performing complex multi-stage maneuvers. Such maneuvers, required for parking and similar situations, are managed in our vehicle as a separate case, as we mention earlier. For scenarios like those, when the car runs at a low speed, a different and more time-consuming approach is used ([8]). The fast CVM-Car is only executed in the follow-lane behavior when the car has to drive at higher velocities. In our approach, the executive layer decides what kind of scenario the car is navigating and selects the appropriate method to use. In this way, the time-consuming convolution between the car footprint and the complex maneuver is avoided at higher speeds.

The final trajectory followed by CVM-Car is similar to the one traveled by a human because of its preference to drive in the longest non-collision arcs (figure 14a). The main difference is that CVM-Car only decides on the basis of the instantaneous situation (current step) and humans can base their decision on future steps. For example, before taking a curve to the right, a race driver will get close to the left side of the lane first to make it smoother. This behavior might be obtained by taking into account several predicted steps in the future for the optimization step. However, that will entail a huge increase in processing time [35].

Finally, any other method derived from CVM, such as LCM or BCM, could have been used instead of CVM. Dynamic obstacles can also be included using BCM-DO [36]. However, a thorough analysis should be carried out to evaluate whether it is worth increasing the complexity of the solution due to the restricted nature of the traffic lane navigation problem.

ACKNOWLEDGMENT

The authors thank all the people that have influenced this work. In particular the researchers of the University of Alcalá ROBOSAFE team that carry out the tests on the electric TABBY EVO car.

REFERENCES

[1] G. Ros, A. Sappa, D. Ponsa, and A. M. Lopez, "Visual Slam for driverless cars: A brief survey," in *Proc. Intell. Vehicles Symp. (IV) Workshops*, vol. 2, 2012, pp. 1–6.

[2] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[3] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments: Part I," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2008, pp. 1063–1069.

[4] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 4, Apr. 1996, pp. 3375–3382.

[5] J. L. Fernández, R. Sanz, J. A. Benayas, and A. R. Diéguez, "Improving collision avoidance for mobile robots in partially known environments: The beam curvature method," *Robot. Auton. Syst.*, vol. 46, no. 4, pp. 205–219, Apr. 2004.

[6] J. López, D. Pérez, and E. Zalama, "A framework for building mobile single and multi-robot applications," *Robot. Auton. Syst.*, vol. 59, nos. 3–4, pp. 151–162, Mar. 2011.

[7] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments: Part II," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2008, pp. 1070–1076.

[8] R. Samaniego, J. Lopez, and F. Vazquez, "Path planning for non-circular, non-holonomic robots in highly cluttered environments," *Sensors*, vol. 17, no. 8, p. 1876, Aug. 2017.

[9] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Robot. Inst., Carnegie-Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. cMU-RI-TR-92-01, Jan. 1992.

[10] J. Lopez, C. Otero, R. Sanz, E. Paz, E. Molinos, and R. Barea, "A new approach to local navigation for autonomous driving vehicles based on the curvature velocity method," in *Proc. Int. Conf. Robot. Automat. (ICRA)*, May 2019, pp. 1752–1757.

[11] C. Otero, R. Sanz, E. Paz, and E. A. J. López, "Simulación de vehículos autónomos usando v-rep bajo ros," in *Proc. 38th Jornadas Automática*, 2017, pp. 806–813.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. Robot Learn.* PMLR, 2017, pp. 1–16.

[13] P. Beeson, J. O'Quin, B. Gillan, T. Nimmagadda, M. Ristroph, D. Li, and P. Stone, "Multiagent interactions in urban driving," *J. Phys. Agents*, vol. 2, no. 1, pp. 15–29, 2008.

[14] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *Int. J. Robot. Res.*, vol. 26, no. 2, pp. 141–166, Feb. 2007.

[15] X. Ji, X. He, C. Lv, Y. Liu, and J. Wu, "Adaptive-neural-network-based robust lateral motion control for autonomous vehicle at driving limits," *Control Eng. Pract.*, vol. 76, pp. 41–53, Jul. 2018.

[16] K. J. Hunt, T. A. Johansen, J. Kalkkuhl, H. Fritz, and T. Gottsche, "Speed control design for an experimental vehicle using a generalized gain scheduling approach," *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 3, pp. 381–395, May 2000.

[17] D. Anderson, "Splined speed control using spam (speed-based acceleration maps) for an autonomous ground vehicle," Ph.D. dissertation, Dept. Mech. Eng., Virginia Tech, Blacksburg, VA, USA, 2008.

[18] F. Cabello, A. Acuña, P. Vallejos, M. E. Orchard, and J. R. del Solar, "Design and validation of a fuzzy longitudinal controller based on a vehicle dynamic simulator," in *Proc. 9th IEEE Int. Conf. Control Automat. (ICCA)*, Dec. 2011, pp. 997–1002.

[19] D. F. Llorca, V. Milanés, I. P. Alonso, M. Gavilán, I. G. Daza, J. Pérez, and M. Á. Sotelo, "Autonomous pedestrian collision avoidance using a fuzzy steering controller," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 390–401, Jun. 2011.

[20] M. Zhu, H. Chen, and G. Xiong, "A model predictive speed tracking control approach for autonomous ground vehicles," *Mech. Syst. Signal Process.*, vol. 87, pp. 138–152, Mar. 2017.

[21] K. Zhang, J. Sprinkle, and R. G. Sanfelice, "Computationally aware control of autonomous vehicles: A hybrid model predictive control approach," *Auton. Robots*, vol. 39, no. 4, pp. 503–517, Dec. 2015.

[22] X. Li, Z. Sun, D. Cao, D. Liu, and H. He, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles," *Mech. Syst. Signal Process.*, vol. 87, pp. 118–137, Mar. 2017.

[23] J. Xue, C. Xia, and J. Zou, "A velocity control strategy for collision avoidance of autonomous agricultural vehicles," *Auton. Robots*, vol. 44, no. 6, pp. 1047–1063, Jul. 2020.

[24] Y. Zhang, H. Chen, S. L. Waslander, J. Gong, G. Xiong, T. Yang, and K. Liu, "Hybrid trajectory planning for autonomous driving in highly constrained environments," *IEEE Access*, vol. 6, pp. 32800–32819, 2018.

[25] S. Li, G. Wang, B. Zhang, Z. Yu, and G. Cui, "Vehicle stability control based on model predictive control considering the changing trend of tire force over the prediction horizon," *IEEE Access*, vol. 7, pp. 6877–6888, 2019.

[26] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 952–964, Feb. 2017.

[27] R. Quirynen, K. Berntorp, K. Kambam, and S. Di Cairano, "Integrated obstacle detection and avoidance in motion planning and predictive control of autonomous vehicles," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2020, pp. 1203–1208.

[28] K. Lee and D. Kum, "Collision avoidance/mitigation system: Motion planning of autonomous vehicle via predictive occupancy map," *IEEE Access*, vol. 7, pp. 52846–52857, 2019.

[29] C. Shi, T. Liu, and Z. Tang, "Motion planning by adding geometric constraint of roadside to beam curvature method," in *Proc. IEEE 3rd Int. Conf. Cyber Technol. Automat., Control Intell. Syst. (CYBER)*, May 2013, pp. 390–395.

[30] L. Qijun and Z. Xiuyan, "An improved BCM obstacle avoidance algorithm for outdoor patrol robot," in *Proc. 6th Int. Conf. Measuring Technol. Mechatronics Automat. (ICMTMA)*, Jan. 2014, pp. 87–90.

[31] J. Benayas, J. Fernández, R. Sanz, and A. Diéguez, "The beam-curvature method: A new approach for improving local realtime obstacle avoidance," *IFAC Proc. Volumes*, vol. 35, no. 1, pp. 409–414, 2002.

[32] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2014, pp. 420–425.

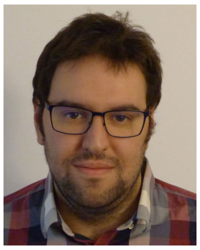
[33] J. López, P. Sánchez-Vilariño, R. Sanz, and E. Paz, "Implementing autonomous driving behaviors using a message driven Petri net framework," *Sensors*, vol. 20, no. 2, p. 449, Jan. 2020.

[34] P. Droździel, S. Tarkowski, I. Rybicka, and R. Wrona, "Drivers 'reaction time research in the conditions in the real traffic," *Open Eng.*, vol. 10, no. 1, pp. 35–47, Jan. 2020.

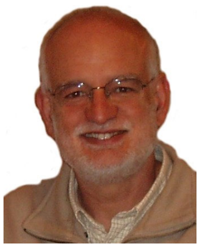
[35] M. Brown, J. Funke, S. Erlien, and J. C. Gerdes, "Safe driving envelopes for path tracking in autonomous vehicles," *Control Eng. Pract.*, vol. 61, pp. 307–316, Apr. 2017.

[36] J. López, P. Sanchez-Vilariño, M. D. Cacho, and E. L. Guillén, "Obstacle avoidance in dynamic environments based on velocity space optimization," *Robot. Auton. Syst.*, vol. 131, Sep. 2020, Art. no. 103569.

JOAQUÍN LÓPEZ received the M.S. degree in telecommunications engineering from the University of Vigo, Spain, in 1992, and the Ph.D. degree from the Department of Systems Engineering and Automation, University of Vigo, in 2000. He spent two years as a Visiting Researcher (first year) and a Special Faculty (NAS2-99020) with the Carnegie Mellon University's Robotics Institute. He is currently an Associate Professor with the School of Industrial Engineering, University of Vigo. He is the author or coauthor of over 45 articles in the fields of mobile robotics and artificial intelligence. He has participated in several funded research projects during the last 15 years.



PABLO SÁNCHEZ-VILARIÑO received the M.S. degree in electrical engineering from the University of Vigo, in 2016. He is currently a Research and Development Engineer with Imatia Innovation, Vigo, Spain. His work focuses on mobile robot control and mobile robots applications. He is also working on several research and development projects focusing on LiDAR technology and computer vision.



RAFAEL SANZ received the degree in electrical engineering from the University of Seville, in 1981, and the Ph.D. degree in electrical engineering from the University of Santiago de Compostela, in 1987. He was a Predoctoral Fellow at L.A.A.S., Toulouse, France. Since 1981, he has been with the Department System Engineering and Automatic Control. He was a Visiting Professor with the University of Western Sydney, Australia, from November 2013 to May 2014. He is currently a Full Professor in system engineering and automation with the University of Vigo, Spain. He has involved in several research and development projects related to the applications of AI in control and mobile robotics. He has published more than 30 articles in robotics and automation. His primary research interests are mobile robotics and intelligent autonomous systems.

ENRIQUE PAZ received the M.S. degree in electrical engineering from the University of Vigo, Spain, in 1991, and the Ph.D. degree from the Department of Systems Engineering and Automation, University of Vigo, in 1997. He is currently an Associate Professor with the School of Industrial Engineering, University of Vigo. His work focuses on mobile robots applications, computer vision, and automation. He has participated in several funded research projects during the last 20 years.