Anticipative kinodynamic motion planner for computing the best path and velocity trajectory in autonomous driving

Jordi Pérez Talamino[a], Alberto Sanfeliu[a]

[a]*Institut de Robòtica i Informàtica Industrial, (CSIC-UPC).*
Parc Tecnològic de Barcelona, 08028, Barcelona, Spain
URL: http://www.iri.upc.edu/

Abstract

This paper presents an approach, using an anticipative kinodynamic motion planner, for obtaining the best trajectory and velocity profile for autonomous driving in dynamic complex environments, such as driving in urban scenarios. The planner discretizes the road search space and looks for the best vehicle path and velocity profile at each control period of time, assuming that the static and dynamic objects have been detected. The main contributions of the work are a fast method for obtaining the G^2-splines for path generation, and a method to compute and select the best velocity profile at each candidate path that fulfills the vehicle kinodynamic constraints, taking into account the passenger comfort. The method has been developed and tested in MATLAB through a set of simulations in different representative scenarios, involving fixed obstacles and moving vehicles. The outcome of the simulations shows that the anticipative kinodynamic planner performs correctly in diverse dynamic scenarios, maintaining smooth accelerations for passenger comfort.

Keywords:
autonomous driving, ADAS, urban, anticipation, kinodynamic motion planning, path planning, G^2-splines, velocity profiles

1. Introduction

Every year 1,25 million people die in traffic accidents worldwide, according to the World Health Organization reports. Taking as an example Europe, road traffic injuries are the leading cause of death among young people. In addition, almost all of the accidents are attributed to driver error, legally intoxicated drivers or distractions. Autonomous vehicles have the potential to dramatically reduce the contribution of negligence and human error as the causes of traffic accidents. Furthermore, they can provide personal mobility to people who are unable to drive because of a disability.

Nevertheless, to achieve this objectives autonomous vehicles must reach a human-acceptable driving performance. This performance level is specially related on developing and improving methods and algorithms that meet the requirements, being crucial motion planning.

We explain in section 3 of this article the motion planner framework that is used; in section 4, the G^2-splines method for path generation; in section 5, the velocity profile generation algorithm and in section 6 the cost structure that is used to select the best path and velocity profile. Finally in section 7 we detail some representative simulations and in section 8, the conclusions. The present article is an extension of the article [1].

Email addresses: joperez@iri.upc.edu (Jordi Pérez Talamino), sanfeliu@iri.upc.edu (Alberto Sanfeliu)

2. Related work

Motion planning for autonomous vehicles requires rapidity (fast decision making), coherency (decisions well aligned with a long term goal), providentness (need to have some temporal horizon to predict) and predictability (follow driving rules and a predefined behaviour) [2]. In order to fulfill those requirements, nowadays exact algorithms with practical computational complexity are unavailable. The most popular methods, state lattices planners, try to discretize somehow the working space (depending on how it is modelled) and search for the best solution [3] [4]. As an example, that kind of framework was chosen by the winners of the DARPA challenges, Stanley in 2005 and BOSS in 2007 [5] [6]. Other planning methods, such as predictive constraint-based planning and spline-based have been proposed [7].

In order to be able to anticipate moving obstacles such as other vehicles, motion planners need a reliable detection of the obstacles and the future behaviour or movement of these vehicles must be imparted or inferred. In general, this prediction problem is extremely difficult. However, when these dynamic obstacles are vehicles operating in urban environments, it can be much easier to infer their likely behaviour through exploiting the structure inherent in such environments: vehicles travelling on roads typically follow common rules of the road [8] [9].

As explained in [4], planners take into account moving obstacles and traffic using different approaches, from Markov Decision Processes, state machines, cost structures, probabilistic models, or Game Theory as an examples.

Motion planners need to generate candidate geometric paths that will be followed by a low level control system. Paths for autonomous driving need to fulfill kinematic and dynamic constraints. In addition, for driving in road scenarios (structured environments), paths need to follow also the road shape. These kinodynamic constraints are mandatory to achieve good control performance and also to assure that paths can be exactly followed by the vehicle. More specifically, these constraints are related with the geometric continuity (G^n) of the path. The most popular used techniques in path generation use arcs, clothoids, polynomial spirals, splines and Bézier curves. They differ in the number of parameters and degrees of freedom, and consequently, there is a trade-off between the level of complexity and performance. Usually the required high level of complexity leads to the presence of non-intuitive geometric waypoints and multitude of parameters that must be adjusted with an important computation load. In [10] paths with continuous rate of curvature are generated using curvature polynomials and a numerical optimization. Method in [11] generates paths that follow the road shape using Bézier curves. Nevertheless, the generation algorithm has limitations due to geometrical constraints and its complexity. Work in [12] proposes a generalization of the used quintic G^2-splines, leading to the G^3-splines, which are 7th order polynomial splines that add curvature derivative continuity.

On the other hand, generating a velocity profile can lead to high order polynomials and complexity, implying high computational cost. In the same way as in the path generation algorithms, the imposed restrictions are related with the function parameters, and often this parameters are not easy to adjust and they need some post-optimization. Velocity profile generation methods often use trapezoidal profiles due to their simplicity despite its dynamic limitations because they have discontinuities in the acceleration [6] [13]. Better approaches compute spline-based velocity trajectories over time, fixing the total maneuver time [14] [15] [16] or also over the road length [17]. Nevertheless, these time conditions make the discretization process difficult and lead even to the need of a post-optimization. Other approaches, such as [13], generate a velocity profile simulating the output of a controller in a certain time horizon taking into account the environment.

3. Anticipative kinodynamic motion planner framework

Fig. 1 shows the general scheme of the proposed motion planner. It is supposed that a route planner computes the desired route to a certain destination. The motion planner needs the paths of the track center lanes, which can be computed with the same path generation algorithm that the planner internally uses, explained in section 4.1, using information from road network data and the perception system. From the perception systems, the planner needs a list with all the detected obstacles, static and dynamic.

Fig. 2 depicts the internal stages of the motion planner. First of all, the planner discretizes the environment choosing several endpoints, which are state configurations $\mathbf{X} = [x, y, \theta, \kappa]$, in a similar way as in [17]. Variables x and y are the position

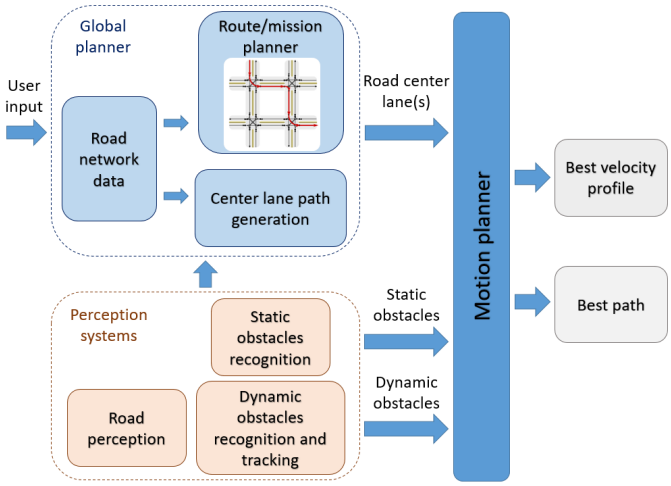


Figure 1: Motion planner framework

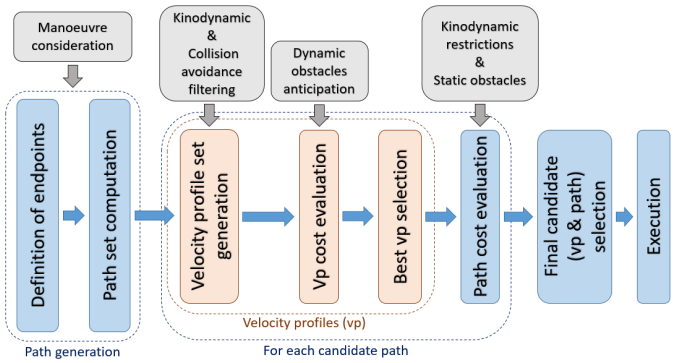


Figure 2: Motion planner detailed operation

coordinates, θ is the heading and κ is the curvature, which is related with the steering wheel angle using the bicycle kinematic model.

Then candidate paths are generated from the host vehicle state to the endpoints (section 4). The motion planner uses a fast approach to generate paths, G^2-splines [18]. This method only needs a basic geometric state $\mathbf{X}$ for the initial and the final points. It can approximate any kind of path shape preserving a smooth curvature continuity and minimizing curvature variability, and this implies kinematic feasibility for a vehicle following it: circular segments, straight lines, clothoids, complete lane changes, etc.

Once the planner has a set of candidate paths, for each one of them it computes a set of velocity profiles in order to anticipate dynamic obstacles (section 5). Each velocity profile has an associated cost and the one with the minimum cost is chosen, for a certain path. Finally, all the candidate paths with its associated velocity profiles are compared with a cost structure (section 6) and the minimum cost solution (path and velocity profile) is chosen to be the executed one. The velocity profiles are computed using 3rd order splines, taking into account the dynamic restrictions of the candidate path and the road path shape, and they are also kinematically validated.

The output of the proposed motion planner is a kinematically

and dynamically feasible path with an associated velocity profile for the host vehicle, also fulfilling comfort restrictions for the passengers, such as bounded accelerations and jerk.

4. Path generation

4.1. G^2-splines path generation

The *G^2-splines* are geometric polynomials of 5th order presenting second order geometric continuity (G^2), so the curvature κ is continuous [18]. In order to build a general path $\mathbf{p}(u)$, understood as a path with arbitrary defined starting endpoint $\mathbf{X}_A = [x_A, y_A, \theta_A, \kappa_A]$, and ending endpoint $\mathbf{X}_B = [x_B, y_B, \theta_B, \kappa_B]$, the equations to define these splines are:

$$\mathbf{p}(u) = \begin{bmatrix} x(u) \\ y(u) \end{bmatrix} := \begin{bmatrix} x_0 + x_1 u + x_2 u^2 + x_3 u^3 + x_4 u^4 + x_5 u^5 \\ y_0 + y_1 u + y_2 u^2 + y_3 u^3 + y_4 u^4 + y_5 u^5 \end{bmatrix} \tag{1}$$

where $u \in [0, 1]$ and:

$$\begin{aligned}
x_0 &= x_A \\
x_1 &= \eta_1 \cos\theta_A \\
x_2 &= \frac{1}{2}(\eta_3 \cos\theta_A - \eta_1^2 \kappa_A \sin\theta_A) \\
x_3 &= 10(x_B - x_A) - (6\eta_1 + \frac{3}{2}\eta_3)\cos\theta_A - (4\eta_2 - \frac{1}{2}\eta_4)\cos\theta_B + \\
&\quad + \frac{3}{2}\eta_1^2 \kappa_A \sin\theta_A - \frac{1}{2}\eta_2^2 \kappa_B \sin\theta_B \\
x_4 &= -15(x_B - x_A) + (8\eta_1 + \frac{3}{2}\eta_3)\cos\theta_A + (7\eta_2 - \eta_4)\cos\theta_B - \\
&\quad - \frac{3}{2}\eta_1^2 \kappa_A \sin\theta_A + \eta_2^2 \kappa_B \sin\theta_B \\
x_5 &= 6(x_B - x_A) - (3\eta_1 + \frac{1}{2}\eta_3)\cos\theta_A - (3\eta_2 - \frac{1}{2}\eta_4)\cos\theta_B + \\
&\quad + \frac{1}{2}\eta_1^2 \kappa_A \sin\theta_A - \frac{1}{2}\eta_2^2 \kappa_B \sin\theta_B \\
y_0 &= y_A \\
y_1 &= \eta_1 \sin\theta_A \\
y_2 &= \frac{1}{2}(\eta_3 \sin\theta_A + \eta_1^2 \kappa_A \cos\theta_A) \\
y_3 &= 10(y_B - y_A) - (6\eta_1 + \frac{3}{2}\eta_3)\sin\theta_A - (4\eta_2 - \frac{1}{2}\eta_4)\sin\theta_B - \\
&\quad - \frac{3}{2}\eta_1^2 \kappa_A \cos\theta_A + \frac{1}{2}\eta_2^2 \kappa_B \cos\theta_B \\
y_4 &= -15(y_B - y_A) + (8\eta_1 + \frac{3}{2}\eta_3)\sin\theta_A + (7\eta_2 - \eta_4)\sin\theta_B + \\
&\quad + \frac{3}{2}\eta_1^2 \kappa_A \cos\theta_A - \eta_2^2 \kappa_B \cos\theta_B \\
y_5 &= 6(y_B - y_A) - (3\eta_1 + \frac{1}{2}\eta_3)\sin\theta_A - (3\eta_2 - \frac{1}{2}\eta_4)\sin\theta_B - \\
&\quad - \frac{1}{2}\eta_1^2 \kappa_A \cos\theta_A + \frac{1}{2}\eta_2^2 \kappa_B \cos\theta_B
\end{aligned}$$

The resulting path depends on the parameter vector $\boldsymbol{\eta}$ that affect its shape, $\boldsymbol{\eta} = [\eta_1, \eta_2, \eta_3, \eta_4]$.

An important characteristic of the resulting spline is its curvature $\kappa(u)$, which can be computed using the following equation:

$$\kappa(u) = \frac{\dot{x}(u)\ddot{y}(u) - \ddot{x}(u)\dot{y}(u)}{(\dot{x}(u)^2 + \dot{y}(u)^2)^{3/2}} \tag{2}$$

4.2. G^2-splines optimization algorithm

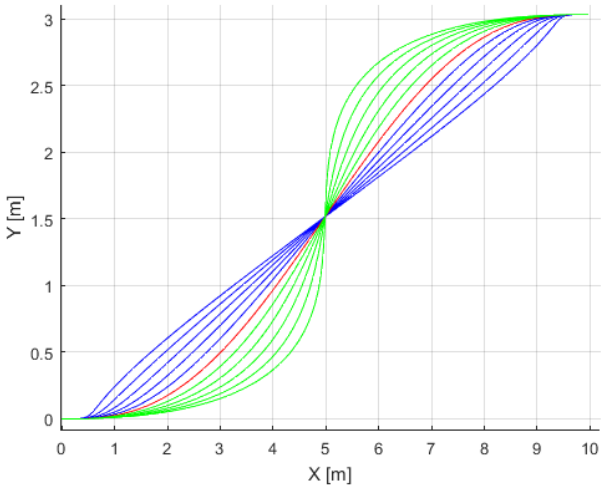
The general case for the optimization of the $\boldsymbol{\eta}$ values requires numerical optimization [18]. However, due to the type of the required vehicle motion maneuvers, we can apply a faster solution that does not need this numerical optimization, but a short iterative process. We have realized that these trajectories have symmetrical behaviour between the starting and ending points, as it can be seen in Fig. 3a, showing a common lane change maneuver. Then we can impose the constraints of Eq. 3 to the $\boldsymbol{\eta}$ parameters, so only 2 parameters are needed to be tuned.

$$\begin{aligned}
\eta_{1_2} &= \eta_1 = \eta_2 \\
\eta_{3_4} &= \eta_3 = -\eta_4
\end{aligned} \tag{3}$$

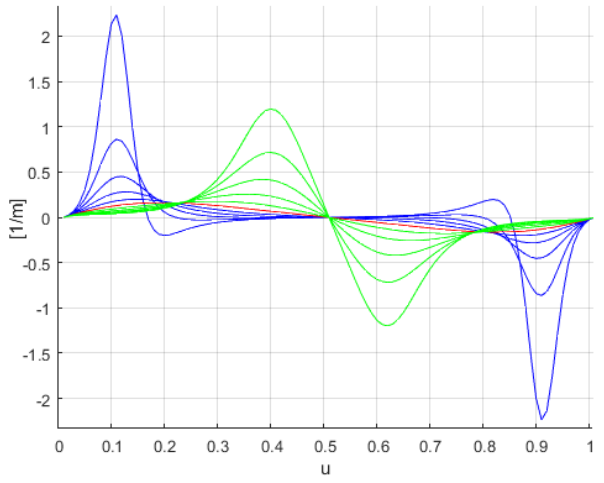
$\eta_{3\_4} \in (-\infty, +\infty)$, and it affects to the curvature changes. It can be concluded that the best value for this parameter is 0 in all the cases. In Fig. 3, different η_{3_4} values are tested performing a lane change maneuver. As it can be seen, if $\eta_{3\_4} > 0$ (green trajectories) the curvature changes are concentrated in the middle part of the trajectory, whereas $\eta_{3\_4} < 0$ (blue trajectories) concentrate it on the extreme initial and final points. The red trajectory represents $\eta_{3\_4} = 0$, and it is the smoothest and more balanced trajectory, presenting the minimum curvature variability.

On the other hand, $\eta_{1\_2} \in (0, +\infty)$, and it forces θ and κ to stay close to the initial and final values. It has been found that the smoothest trajectory is reached when η_{1_2} is close to the trajectory length (in meters). For this reason, an iterative method is performed in order to converge into the best possible trajectory, giving to η_{1_2} the length of the path. After a few iterations, usually 3 or 4, all the computed trajectories converge into the smoothest one (Fig. 4). This process obtains similar results as the proposed numerical optimization in [18], which minimizes the curvature variability $\left(\min_{\boldsymbol{\eta}} \frac{d\kappa}{du}\right)$, but with much less computation allowing to reach real time performance.

Fig. 5 shows a comparison between 1 and 5 iterations of the optimization algorithm for generating the center lane paths of a complex intersection. The paths are generated only with the geometrical information $\mathbf{X}$ of the initial and final points.

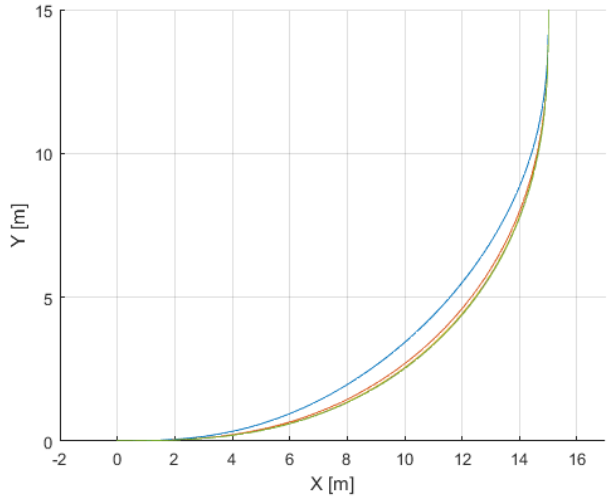


(a) Path X-Y position, in $[m]$

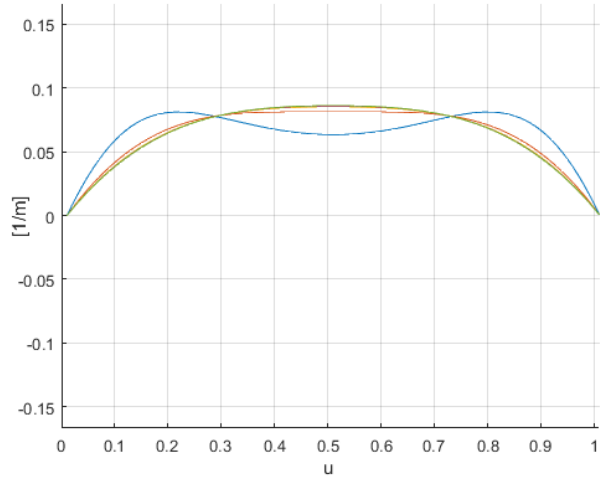


(b) Path curvature, in $[1/m]$

Figure 3: η_{3_4} variations in a lane change maneuver



(a) Path X-Y position, in $[m]$



(b) Path curvature, in $[1/m]$

Figure 4: η_{1_2} iterations approximating a circular path. 1st iteration in blue, 2nd in orange and 3rd in green

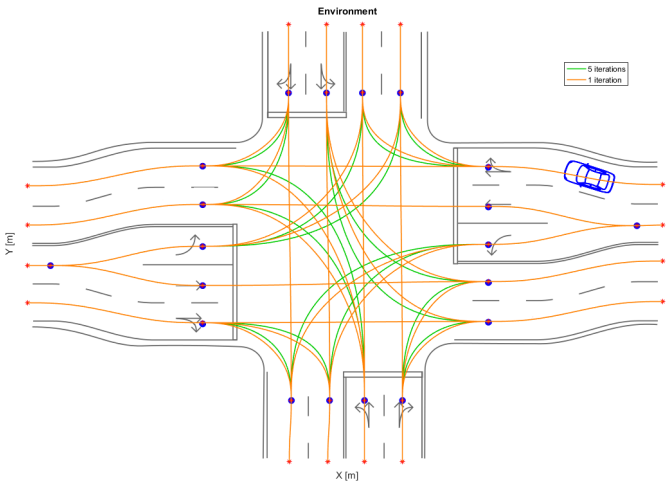


Figure 5: Center lane paths generated with the G^2-splines algorithm in a complex intersection, with different iterations (1 in orange and 5 in green). The only used information is the endpoints x, y and θ, as all the endpoints curvatures are considered as 0

5. Velocity profile generation

To generate the velocity profiles, the motion planner uses a 3rd grade polynomial spline in velocity, $v(t)$. This spline gives a smooth transition between two different velocities, and it has a parabolic acceleration profile fixing the maximum (or minimum) acceleration. This smooth and comfortable transition between velocities is caused by the derivative of the acceleration, the jerk, which is continuous.

In addition, the possibility of adjusting directly this parameter of maximum desired acceleration in the trajectory is very useful for this application. It is directly related with the time needed for the trajectory and also with the comfort of the passengers and its feasibility.

The velocity profile generation algorithm is splitted in 2 different situations: with and without initial acceleration, explained below. This allows having always an analytical solution to reduce the computational cost.

5.1. 3rd order spline without initial acceleration

The proposed 3rd order spline in velocity has 4 variables: a, b, c, d (Eq. 4). The fifth unknown is the total time of the trajectory, T. The equation system (Eq. 5) is solved by applying initial and final conditions, and also using the fact that the velocity spline is symmetric, so the maximum acceleration is achieved at time equal to $T/2$.

$$\begin{aligned} x(t) &= \tfrac{a}{4}t^4 + \tfrac{b}{3}t^3 + \tfrac{c}{2}t^2 + dt + x_0 \\ v(t) &= at^3 + bt^2 + ct + d \\ a(t) &= 3at^2 + 2bt + c \end{aligned} \tag{4}$$

$$\begin{cases} v(0) = v_0 \\ a(0) = 0 \\ v(T) = v_f \\ a(T) = 0 \\ a(\frac{T}{2}) = a_{max} \end{cases} \quad \begin{cases} b = \dfrac{4a_{max}^2}{3(v_f - v_0)} \\ a = -\dfrac{b^2}{3a_{max}} \\ c = 0 \\ d = v_0 \\ T = \dfrac{3(v_f - v_0)}{2a_{max}} \end{cases} \tag{5}$$

5.2. 3rd order spline with arbitrary initial acceleration

This 3rd order spline in velocity is the same as in the case without initial acceleration (Eq. 4), but the equation system is slightly different because the spline is not symmetric and the maximum acceleration is reached in the time t_1. The 6 unknowns are the spline variables a, b, c, d, the total time of the trajectory T and the time t_1.

$$\begin{cases} v(0) = v_0 \\ a(0) = a_0 \\ v(T) = v_f \\ a(T) = 0 \\ a(t_1) = a_{max} \\ a'(t_1) = 0 \end{cases} \quad \begin{cases} 0 = \left[\dfrac{4a_0^2}{a_0 - a_{max}} - 3a_0\right]T^2 + \\ \quad + \left[6(v_f - v_0) - 12\dfrac{(v_f - v_0)a_0}{a_0 - a_{max}}\right]T + \\ \quad + \left[\frac{9(v_f - v_0)^2}{a_0 - a_{max}}\right] \\ b = \dfrac{1}{T}\left[\frac{3(v_f - v_0)}{T} - 2a_0\right] \\ a = \dfrac{b^2}{3(a_0 - a_{max})} \\ c = a_0 \\ d = v_0 \\ t_1 = -\dfrac{b}{3a} \end{cases} \tag{6}$$

We have to solve a 2nd order equation to find the times T_1 and T_2 (Eq. 6). Not in all cases there exists a solution, due to the denominator term $(a_0 - a_{max})$. a_{max} must be always greater than a_0 if accelerating or lower when decelerating.

When only one of the T_i is positive, then this is the unique solution. In the case when both T_i are positive corresponds to two possible solutions: increasing acceleration up to a_{max} and finishing faster the maneuver, or decreasing the acceleration from a_0 and finishing the maneuver in longer time. In this case, the fastest maneuver is always chosen, so $T = min(T_1, T_2)$.

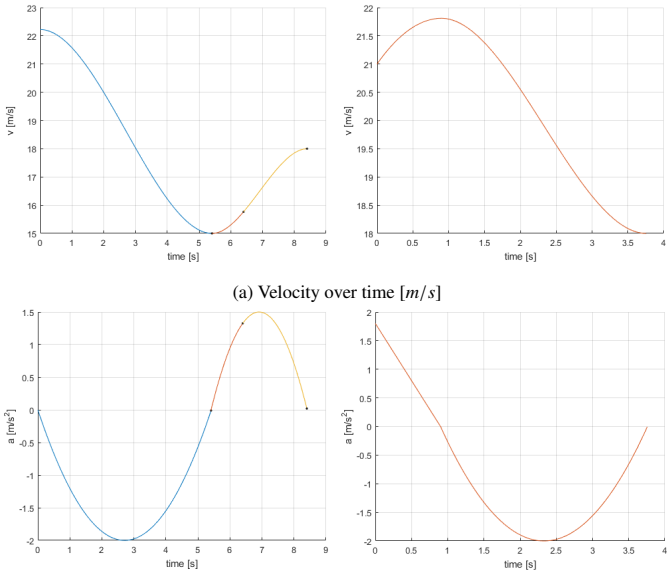
The cases when the spline has no solution correspond to situations when the vehicle is accelerating in the opposite direction of the desired final speed, or *contradictory cases* (for example, with $a_0 > 0$ and $v_f < v_0$). This issue is solved by adding a linear acceleration section that starts in a_0 and ends in 0 acceleration, with a desired slope (jerk) that guarantees comfort and feasibility.

5.3. Velocity profile generation algorithm

The complete algorithm for generating a velocity profile is detailed in Algorithm 1, and some examples are shown in Fig. 6. Each colour is a different launch of the algorithm. The blue curve is the spline without initial acceleration. Then, it is computed a new velocity profile to 18 m/s (in red) but in a certain instant, the algorithm is re-launched to the same final speed (in orange), which corresponds to the spline with arbitrary initial acceleration. The example on the right shows a *contradictory case*.

Algorithm 1 Generate a velocity profile

```
if a_0 = 0 then
    Spline case without a_0
else
    if v_f = v_0 then
        Add linear acceleration from a_0 to 0
        Spline case without a_0
    else
        if contradictory case then
            Add linear acceleration from a_0 to 0
            Spline case without a_0
        else
            Spline case with arbitrary a_0
        end if
    end if
end if
```



(a) Velocity over time $[m/s]$

(b) Acceleration over time $[m/s^2]$

Figure 6: Different launches of the velocity profile generation, in different colours

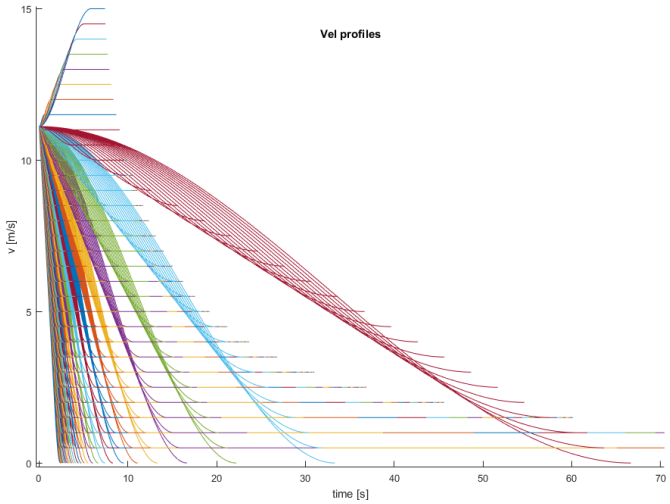
5.4. *Set of velocity profile candidates*

For each candidate path, the motion planner computes a set of velocity profile candidates, discretizing the final velocities and also the accelerations, taking into account the kinodynamics of the vehicle. Starting in the current state of the host vehicle, several final velocities are chosen from 0 to the maximum allowed road/street velocity. Then, for each final velocity, different accelerations are also chosen, from the maximum deceleration limit to the desired acceleration value. Examples of these velocities and accelerations can be seen in Fig. 7.

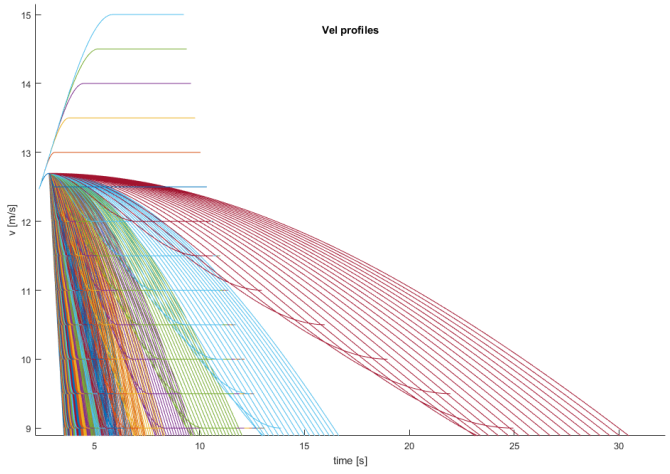
In order to reduce the computation, we impose some constraints in the velocity profile using the splines to reject the non-feasible candidates and restrict the search space. The constraints are:

1. The distance to collide with a static obstacle, by using the distance in length of the track from the host vehicle. If a collision is detected, the only allowed final velocity is 0.
2. The maximum allowed velocity due to the planned path curvature.
3. The maximum allowed velocity due to the center lane path curvature. This restriction assures safety, in order to be more conservative and check a static property of the track geometry, because the planned path could be smoothed when re-launching.
4. The longitudinal dynamics of the vehicle. This restricts the accelerations to ensure feasibility.

With this approach, we can have the required accelerations, and then we can reduce the search space and prune the velocity profile selecting the best ones.



(a) Example with $a_0 = 0\ m/s^2$



(b) Example with $a_0 > 0\ m/s^2$. Profiles with a lower final velocity are *contradictory cases*

Figure 7: Sets of all the generated velocity profiles, with a discretization of 0.5 m/s and different accelerations

6. Selection of the best path and velocity profile

The selection of the motion planner output is made in each iteration by choosing the minimum cost solution according to the total cost function J, which is a weighted sum of all the cost terms (Eq. 7).

$$J = J_{static} + J_{dynamic} = \sum_i w_{s,i} \cdot c_{s,i} + \sum_j w_{d,j} \cdot c_{d,j} \quad (7)$$

The motion planner uses two kind of cost terms: static and dynamic. Static costs are the terms related to path geometry and static obstacles. They are associated to a candidate path (Table 1). In the other hand, dynamic costs are related to the temporal dimension and they are associated to a velocity profile and include the dynamic obstacles (Table 2). The costs are based on the method proposed in [17], but adapted to fit the proposed approach.

Cost	Formula
c_l	$l/s_{maneuver}$
c_κ	$max(\kappa) \cdot r_{min}$
$c_{\dot{\kappa}}$	$max(\dot{\kappa}) \cdot r_{min}$
c_{off}	o/o_{max}
$c_{obs,s}$	$f \cdot e^{(-1/\lambda)\cdot d}$

Table 1: Static costs

Cost	Formula
c_v	$1 - v_{f,vp}/v_{max,desired}$
c_a	$abs(a_{max,vp}/a_{max,braking})$
$c_{obs,d}$	$f \cdot e^{(-1/\lambda)\cdot d}$

Table 2: Dynamic costs

All the terms (static and dynamic) with the exception of the obstacle terms $c_{obs,s}$ and $c_{obs,d}$, are normalized between 0 and 1. Each term is detailed below:

- c_l : Path length term. It has an impact in the efficiency. l is the length of the candidate path. $s_{maneuver}$ is the *station* of the maneuver: the length from the host vehicle to the ending point of the path referred to the road axis.
- c_κ and $c_{\dot{\kappa}}$: Curvature terms. Related to comfort and kinematic feasibility. κ and $\dot{\kappa}$ are the curvature and curvature derivative values respectively of a specific point in the path candidate and r_{min} is the minimum turning radius of the host vehicle.
- c_{off}: Lateral offset term. It has an impact in the behaviour (it tries to drive centered in the lane). o_{max} is the lateral distance from the farthest endpoint of all the paths in the set to the center of the lane (the maximum admissible lateral offset) and o is the lateral offset of the path candidate endpoint.
- c_v: Velocity term. Affects the behaviour (it tries to drive at the maximum allowed speed). $v_{f,vp}$ is the ending velocity of the candidate velocity profile. $v_{max,desired}$ is the maximum allowed velocity at the current moment.
- c_a: Acceleration term. It has an impact in comfort and efficiency. $a_{max,vp}$ is the maximum acceleration value of the candidate velocity profile. $a_{max,braking}$ is the acceleration value (always < 0) of the maximum allowed braking of the host vehicle.
- $c_{obs,s}$ and $c_{obs,d}$: Static and dynamic terms. f is a scale value and λ is the decay of the exponential function. d is the distance from a obstacle (static or dynamic) to the host vehicle (they have been modelled as circles). If d is smaller than a threshold, then the cost is penalized with P:

$$c_{obs} = \begin{cases} f \cdot e^{(-1/\lambda)\cdot d} + P, & if\ d < threshold \\ f \cdot e^{(-1/\lambda)\cdot d}, & otherwise \end{cases} \quad (8)$$

The function, without the penalization term, is shown in Figure 8.

Unlike other approaches that penalize collisions with an infinite cost, as [17], it is preferred to preserve the value. Then, in a situation where all the candidate solutions present high cost (for instance, an unavoidable obstacle), the planner will give always the minimum cost solution, so it will be the less dangerous one. In fact, in this ultimate situations the output is a maximum braking until stopping inside the lane, which is the safest action that a human driver could do.

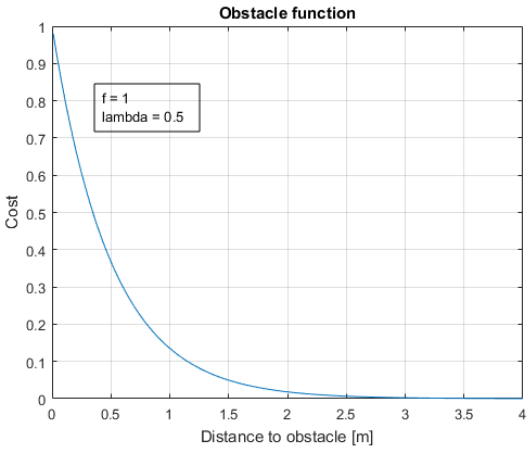


Figure 8: Obstacle cost function c_{obs}, without the penalization term

The chosen velocity profile for each candidate path is the one with minimum dynamic cost $J_{dynamic}$. Then, each candidate path forms a maneuver candidate, as it has a velocity profile associated, and the static cost function J_{static} is computed.

Finally the minimum cost maneuver is the chosen one, which minimizes the total cost function J (Eq. 7).

7. Simulations

7.1. Simulation environment

The motion planner and the low level vehicle control have been implemented in MATLAB. The Stanley's lateral controller [19] has been used together with a longitudinal controller which takes into account the dynamics of the vehicle. More concretely, the steering column has been modelled with a first order system and the used longitudinal model takes into account gear changes and internal combustion engine torque response. The control system has been validated and adjusted through simulation using data from a SEAT Cupra car.

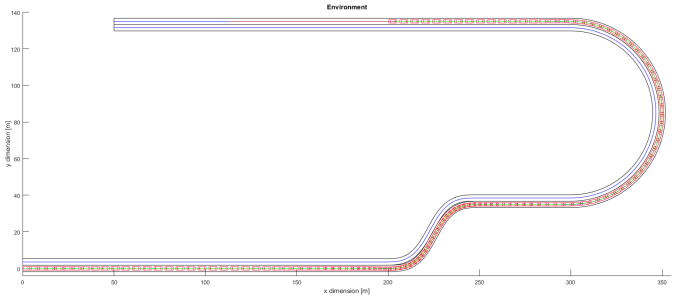
The chosen scenarios have been selected because they represent real-life common situations in urban and road driving, and they are challenging tests to validate the correct behaviour of the proposed motion planner.

Lanes are represented with the border lines (in black) and its center-lines (in blue). The path followed by the host vehicle is shown in green. The host vehicle (in red and purple) is plotted at a regular interval of time. The simulations are performed in MATLAB using a mid-range computer hardware and the average computation for each simulation cycle with static and dynamic obstacles is 250 ms. Taking into account that a real application code in C++ being executed is around 10 times faster, this method can run at 25 ms, even without considering implementation optimizations.

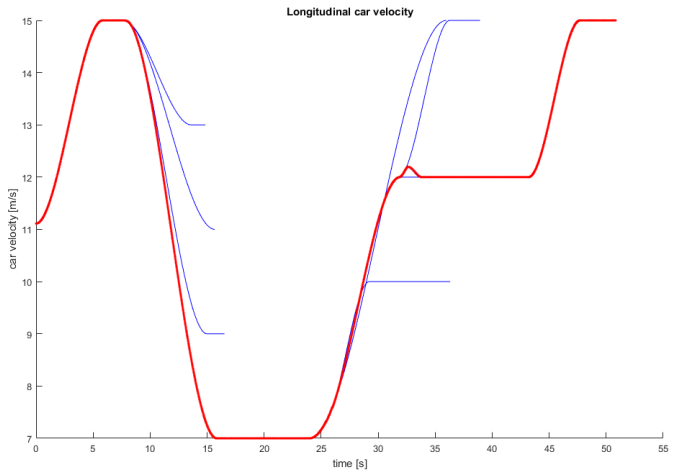
7.2. Static environments

Fig. 9 shows a simulation in a road environment without any obstacle in the lanes. The maximum allowed velocity is 15 m/s, and the host vehicle starts at 11.11 m/s (40 km/h). At the beginning, it accelerates to the maximum speed, until encountering the "S" turn, which is passed at 7 m/s. Then, speed is incremented up to 12 m/s because of the left turn, and in the final straight section the vehicle accelerates again to the maximum allowed speed.

Fig. 10 shows the same environment, but with several static obstacles, represented by circles in the lanes. This simulates an urban scenario, for instance with vehicles parked in double row. The vehicle avoids the obstacles moving inside the lane but also changing lanes.



(a) Followed path. Dimensions in [m]



(b) Vehicle velocity over time (in red) and planned velocity profiles (in blue), in [m/s]

Figure 9: Simulation without any obstacle.

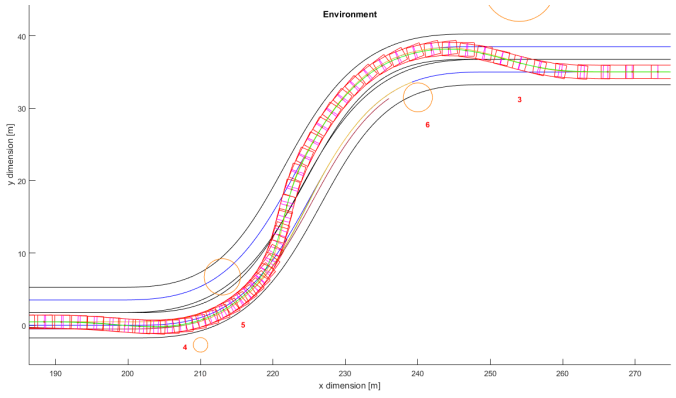


Figure 10: Detail of a static obstacle avoidance maneuver during a "S" turn. Dimensions in [m]

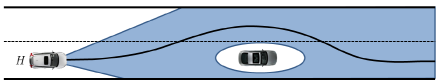
7.3. Dynamic environments

The simulations present in this section take into account the influence of the dynamic obstacles in the near future.

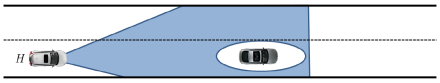
7.3.1. Overtaking in one way track and with oncoming traffic

Next simulations show examples of overtaking. When overtaking, one consideration to be taken into account is the planning horizon. Fig. 11a shows a planning example of a whole overtaking maneuver. In this situation, the planning horizon is assumed to be long enough to allow the complete maneuver planning. The planner can anticipate dynamic obstacles up to the maneuver completion, so if it initiates the overtaking is because it is safe. If the left lane is an oncoming traffic lane, this horizon must be fulfilled in order to guarantee safety.

Now, consider another situation, shown in Fig. 11b. In this case, due to some reason, the perception system range is smaller than the needed horizon for overtaking. What should the planner do? If the other lane has the same way, planning only a lane change maneuver to the other lane is safe enough, because the host vehicle only has to be careful with vehicles coming behind. But in the case of oncoming traffic, the overtaking cannot be performed. If the obstacle is static, the best solution is to stop the host vehicle safely in its lane.



(a) Enough prediction horizon to plan a complete overtaking maneuver



(b) Smaller prediction horizon due to restrictions in the perception range

Figure 11: Planning horizon considerations

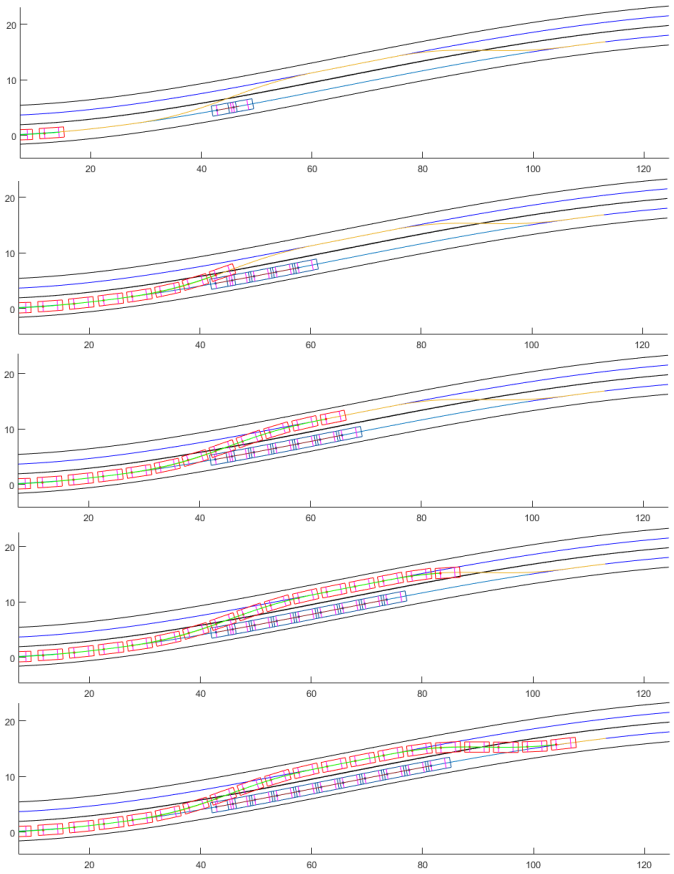
Fig. 12 shows an overtaking maneuver in a one way track (both lanes have the same direction). The host vehicle starts at 14 m/s and tries to reach the maximum speed of 15 m/s, but then it encounters a slower vehicle ahead going at 5 m/s. At first, the motion planner plans a velocity profile to 6,5 m/s because the complete maneuver until the desired horizon of 100 m ahead cannot be done. But when the host vehicle gets closer to the ahead vehicle, the planner finds a feasible overtaking solution.

In the simulation of Fig. 13 the host vehicle is travelling in a 2-way track. It encounters the lane blocked ahead by some static obstacles, and there is an oncoming vehicle in the other lane. The planner slows down smoothly to 1 m/s remaining in the right lane until the oncoming vehicle (in blue) passes by. Then it can cross to the left lane and finish the maneuver.

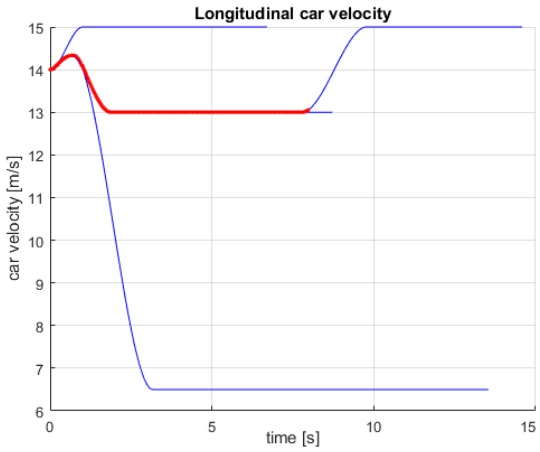
Overtaking in a 2-way track can be done only if the maneuver is predicted to its end, otherwise it is discarded and the host vehicle slows down and follows the dynamic obstacle (the preceding vehicle). Fig. 14 shows a simulation where the host vehicle can take advantage of its initial higher velocity and return to the right lane before the oncoming traffic. However, if the oncoming traffic prevents the manoeuvre, the host vehicle will wait behind. This behaviour can be seen in Fig. 15. Different responses can be obtained adjusting the weights of the cost terms, and also changing the repulsion function parameters of Eq. 8.

7.3.2. T-intersection

To enter in a T-intersection is mandatory to anticipate the oncoming vehicles, which come in both directions. Fig. 16 shows a conservative simulation, where the host vehicle slows down to 2 m/s and let the traffic pass before. On the other hand, Fig. 17 is a more aggressive situation, where the host vehicle takes advantage of a gap between vehicles and enters to the intersection at its maximum allowed speed due to the lateral comfort acceleration, that is 8 m/s.

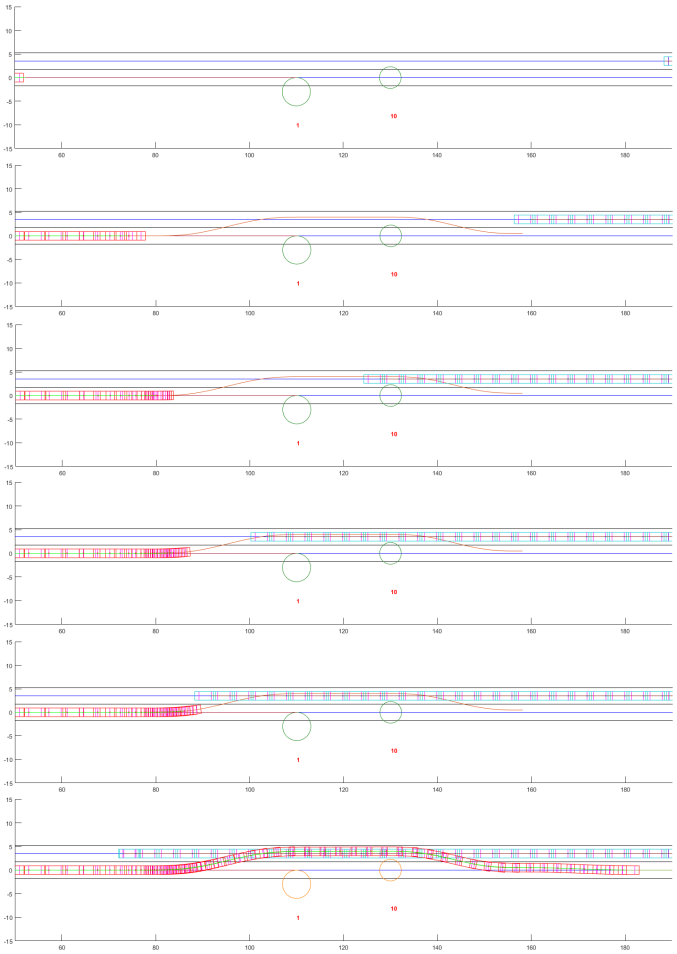


(a) maneuver at different instants. Dimensions in [m]

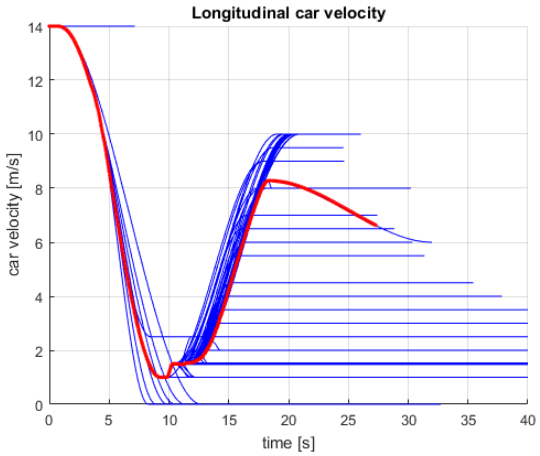


(b) Vehicle velocity over time (in red) and planned velocity profiles (in blue), in $[m/s]$

Figure 12: Overtaking of a slower vehicle in a one way track. Red vehicle is the host.

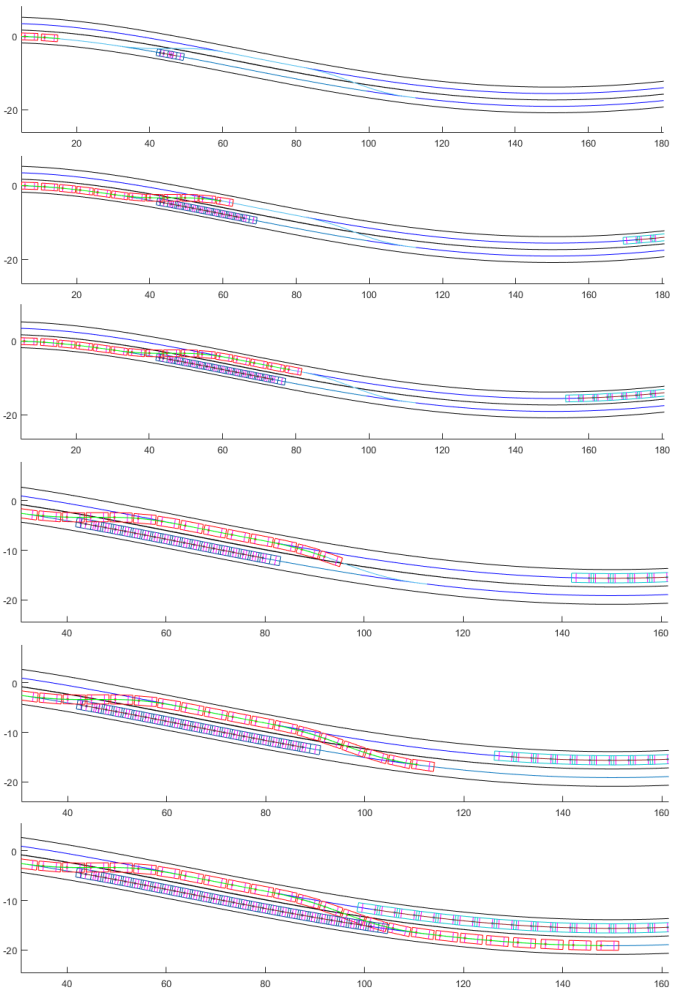


(a) maneuver at different instants. Dimensions in [m]

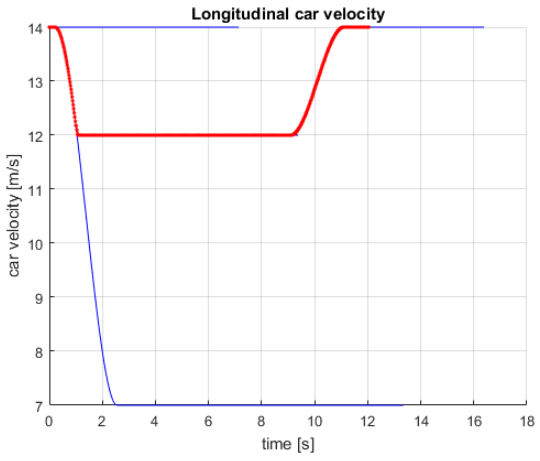


(b) Vehicle velocity over time (in red) and planned velocity profiles (in blue), in $[m/s]$

Figure 13: Overtaking of a static obstacle in a two way track

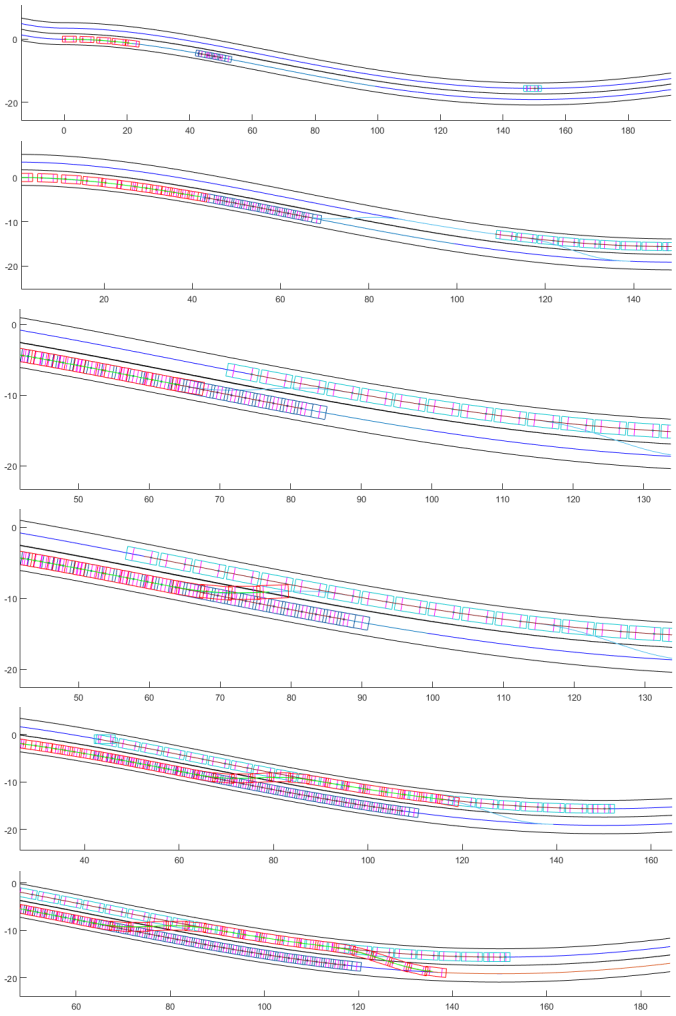


(a) maneuver at different instants. Dimensions in [m]

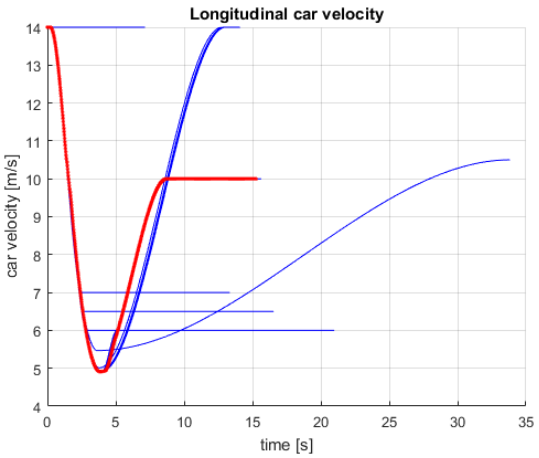


(b) Vehicle velocity over time (in red) and planned velocity profiles (in blue), in $[m/s]$

Figure 14: Overtaking of a slower vehicle in a two way track with oncoming traffic. Red vehicle is the host. Aggressive case

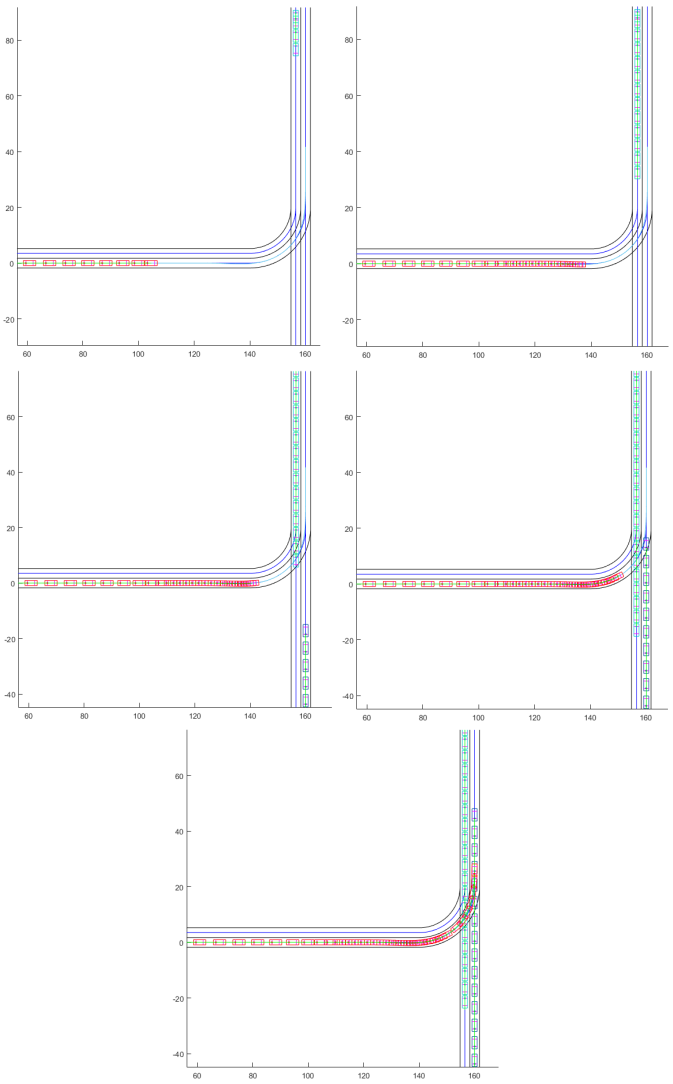


(a) maneuver at different instants. Dimensions in [m]

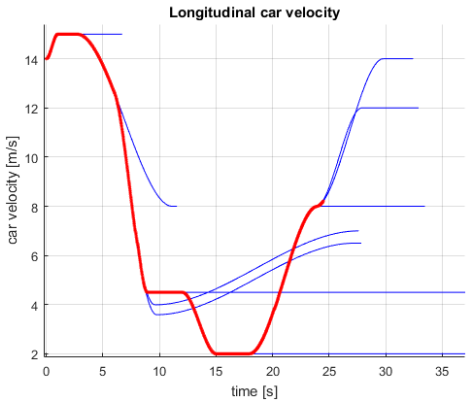


(b) Vehicle velocity over time (in red) and planned velocity profiles (in blue), in $[m/s]$

Figure 15: Overtaking of a slower vehicle in a two way track with oncoming traffic. Red vehicle is the host. Conservative case

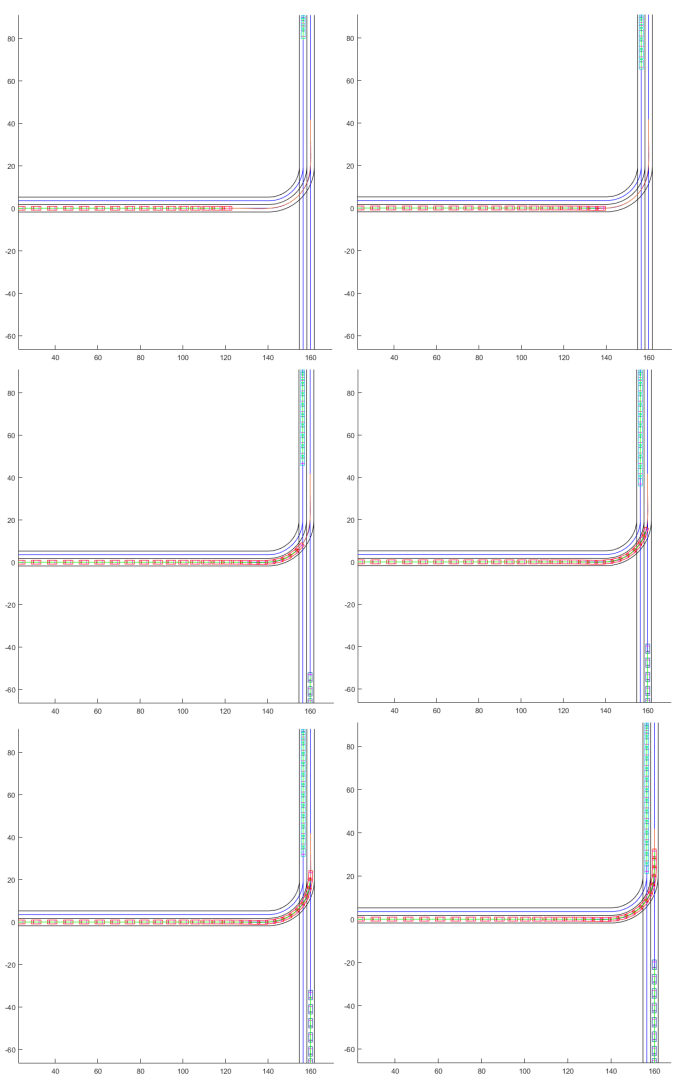


(a) maneuver at different instants. Dimensions in [m]

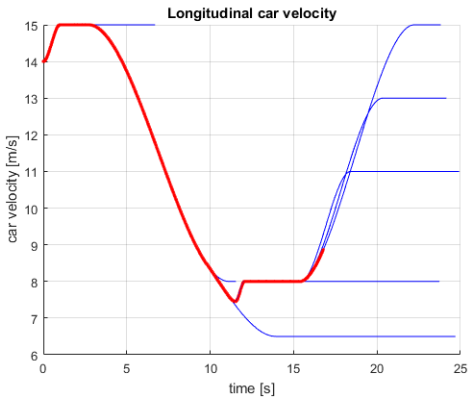


(b) Vehicle velocity over time (in red) and planned velocity profiles (in blue), in $[m/s]$

Figure 16: Incorporation in a T-intersection, in several instants of time. Conservative case



(a) maneuver at different instants. Dimensions in [m]



(b) Vehicle velocity over time (in red) and planned velocity profiles (in blue), in $[m/s]$

Figure 17: Incorporation in a T-intersection, in several instants of time. Aggressive case

8. Conclusions

After analyzing the simulations, it is verified that the proposed methods of path and velocity profile generation behaves correctly and help to overcome successfully all the presented situations, which correspond to representative cases that occur in on-road driving.

The proposed approach for the path generation, using the G^2-splines, performs above expectations and a real-time computation can be obtained. The paths can be generated only with basic information of the initial and final endpoints, which contributes to reduce complexity and improves performance, flexibility and the ease of use of the algorithm.

The proposed method for the velocity profile generation behaves smoothly and provides a fully analytic solution that can deal with any arbitrary situation. Moreover, it allows to directly adjust the desired acceleration instead of the maneuver time and this helps to the selection and pruning of a candidate set of velocity profiles, taking into account kinematic and dynamic constraints.

9. Acknowledgements

Work supported by the Spanish Ministry of Science and Innovation under project ColRobTransp (DPI2016-78957-RAEI/FEDER EU) and by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656).

10. References

[1] J. P. Talamino, A. Sanfeliu, *Path and velocity trajectory selection in an anticipative kinodynamic motion planner for autonomous driving*. ROBOT 2017: Third Iberian Robotics Conference, Advances in Intelligent Systems and Computing 694, pp. 434-445, 2018.

[2] S. Ulbrich, M. Maurer, *Probabilistic Online POMDP Decision Making for Lane Changes in Fully Automated Driving*. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 2063-2067, 2013.

[3] B. Paden, M. Cáp, S. Zheng Yong, D. Yershov, E. Frazzoli, *A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles*. IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pp. 33-55, 2016.

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

[5] S. Thrun, M. Montemerlo *et al.*, *Stanley: The Robot that Won the DARPA Grand Challenge*. Journal of Field Robotics, vol. 23, issue 9, pp. 661-692, 2006.

[6] D. Ferguson, T. M. Howard, M. Likhachev, *Motion Planning in Urban Environments, Part I and II*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1063-1069, 1070-1076, 2008.

[7] D. Madas, M. Nosratinia, M. Keshavarz, *et al.*, *On path planning methods for automotive collision avoidance*. Intelligent Vehicles Symposium (IV) IEEE, pp. 931–937, 2013.

[8] D. Ferguson, M. Darms, C. Urmson, S. Kolski, *Detection, Prediction, and Avoidance of Dynamic Obstacles in Urban Environments*. IEEE Intelligent Vehicles Symposium, pp. 1149-1154, 2008.

[9] J. Wei, J. M. Snider, Tianyu Gu, J. M. Dolan, B. Litkouhi, *A Behavioral Planning Framework for Autonomous Driving*. Intelligent Vehicles Symposium Proceedings, pp. 458-464, 2014.

[10] A. Kelly, B. Nagy, *Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control*. The International Journal of Robotics Research, vol. 22, issue 7-8, pp. 583-601, 2003.

[11] D. G. Bautista, J. P. Rastelli, R. Lattarulo, V. Milanés, F. Nashashibi, *Continuous curvature planning with obstacle avoidance capabilities in urban scenarios*. 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1430-1435, 2014.

[12] A. Piazzi, M. Romano, C. G. Bianco, *G^3-splines for the path planning of wheeled mobile robots*. European Control Conference (ECC), pp. 1845-1850, 2003.

[13] M. McNaughton, *Parallel Algorithms for Real-time Motion Planning*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2011.

[14] T. Gu, J. M. Dolan, *On-Road Motion Planning for Autonomous Vehicles*. International Conference on Intelligent Robotics and Applications (ICIRA), vol. 3, pp. 588-597, 2012.

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

[16] M. Werling, J. Ziegler, S. Kammel, S. Thrun, *Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenét Frame*. IEEE International Conference on Robotics and Automation (ICRA), pp. 987-993, 2010.

[17] W. Xu, J. Wei, J. M. Dolan, H. Zhao, H. Zha, *A Real-Time Motion Planner with Trajectory Optimization for Autonomous Vehicles*. IEEE International Conference on Robotics and Automation (ICRA), pp. 2061-2067, 2012.

[18] C. G. Bianco, A. Piazzi, *Optimal trajectory planning with quintic G^2-splines*. Proceedings of the IEEE Intelligent Vehicles Symposium, pp. 620-625, 2000.

[19] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, S. Thrun, *Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing*. American Control Conference (ACC), pp. 2296-2301, 2007.