Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 98 (2016) 148 – 155

Procedia Computer Science

The 7th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2016)

Lateral Control of an Unmaned Car Using GNSS Positionning in the Context of Connected Vehicles

Alexandre Lombard[a,*], Xuguang Hao[a], Abdeljalil Abbas-Turki[a], Abdellah El Moudni[b], Stéphane Galland[a], Ansar-Ul-Haque Yasar[c]

[a] *Université Bourgogne Franche-Comté, UTBM, LE2I UMR CNRS 6306, 90010 Belfort cedex, France*
[b] *Université Bourgogne Franche-Comté, UTBM, Nanomédecine, imagerie, thérapeutique EA 4662, 90010 Belfort cedex, France*
[c] *Transportation Research Institute (IMOB), Hasselt University, Wetenschapspark 5 bus 6, 3590 Diepenbeek, Belgium*

Abstract

This paper addresses the problem of path following for unmaned cars using only GNSS positionning. Several papers have already addressed the lane keeping, usually restricting the focus to the computation of the optimal steering wheel angle, therefore usually offering a good solution in either straight lines or curves, but rarely both. Moreover the GNSS receiver have a sampling time higher than the required one for the proposed strategies in the literature. In this paper, a control algorithm is proposed for the steering wheel angle, with adaptation of the speed of the car, in order to provide a solution working both for straight lines and curves by using only GNSS. The presented solution has been used in real conditions to realize the demonstration "speed synchronization at intersections using intervehicular communication, x.icars" shown at the Intelligent Transportation System World Congress in Bordeaux, 2015 [1].

Peer-review under responsibility of the Program Chairs

Keywords: autonomous vehicles, gnss positionning, direction control, acceleration control

1. Introduction

In the x.icars demonstration, there are three driverless vehicles that share an 8-shaped circuit. All main sensors are disabled. They rely only on wireless communication (IEEE 802.11p) and positioning system to move forward and to avoid collision. There are three important control issues that are successfully considered:

- Longitudinal control, to take into consideration the frontal obstacle. This control considers the communication delay.
- Lateral control to keep the lane.

[1] https://www.youtube.com/watch?v=KBx6qks8kCI

* Corresponding author. Tel.: +33-384-583-431.
E-mail address: alexandre.lombard@utbm.fr

Peer-review under responsibility of the Program Chairs
doi:10.1016/j.procs.2016.09.023

- Intersection control where a first-in first-out (FIFO) policy is used (see [1]).

This paper focuses on lateral control. Path tracking is an important research subject of robotic, and several algorithms have designed in order to achieve this goal. The objective is to, according to a position given by sensors (radar, lidar, GPS, cameras, etc.), compute the adequate control to follow a pre-defined track. Path tracking applications are numerous and includes Automated Guided Vehicles (AGV) and autonomous cars among others.

In this paper we present an overview of the literature to brievly present existing solution to perform path tracking and their drawbacks, then we propose a new solution for the path tracking problem, using only GNSS positionning, for a robotic car, combining wheel angle control and acceleration control to get a good accuracy on the trajectory.

2. Litterature overview

The path-following problem is a well-known problem and several solutions exist in order to make a robot, or a car, follow a given track. The lateral control problem is a nonlinear control problem. There are many proposed approaches. This section presents an overview of the mainly used solutions according to [2] and [3] for the lateral control.

2.1. Pure pursuit

In the pure-pursuit algorithm [4], the robot knows its position, then it projects it on the reference track and define on this track a futher point to follow at a constant look-ahead distance. Then the steering wheel angle required to follow this point is computed, and applied.

Major drawbacks of the pure pursuit is that a followed point closed to the vehicle position will induce oscillations in the car's behavior, and that either a point far from the vehicle or an important vehicle speed will prevent the vehicle from following the track in curves.

2.2. Stanley method

The Stanley method [5] is a path tracking approach in which one the wheel angle $\delta(t)$ is defined by (1).

$$\delta(t) = \theta_e(t) + tan^{-1}(\frac{ke_{fa}(t)}{v_x(t)}) \tag{1}$$

With $\theta_e(t)$ being the difference between the difference between the heading of the car and the heading of the path, $e_{fa}(t)$ the distance between the front axle and the path, $v_x(t)$ the speed of the car and k a gain parameter.

Unlike the pure-pursuit, the Stanley method does not suffer from the "cutting corners" effect and is more suitable for a high velocity. However, it still presents a problem of overshooting at the end of the curves, and suffers more from disturbances than the pure pursuit.

2.3. Vector pursuit

Another solution to the path-following problem is the vector pursuit [6], based on the theory of screws. Similarly to the pure pursuit, a look-ahead point is computed but it uses both its position and its orientation (i.e. the tangent to the track at the look-ahead point).

Even though this algorithm performs better than the pure pursuit algorithm, it suffers from similar drawbacks as it ignores the influence of the vehicle speed on the path-tracking performance: a look-ahead point placed too close from the vehicle will still induce oscillations when speed increases, while a look-ahead point placed too far will induce too much anticipation in path tracking (i.e. the car will start turning before a curve).

2.4. Linear quadratic regulator (LQR)

Other works are based on the control theroy and describe the system dynamics by a set of linear differential equations with a cost described by a quadratic function [7]. Then a linear-quadratic regulator is used as a controller to set the steering wheel angle.

Fig. 1: Photograph of the automated Renault Scenic during path-tracking tests on the ITS World Congress 2015 site in Bordeaux

Using a regulator in order to control the direction of a car requires a lot of tuning. Moreover, unlike the pure pursuit based methods, this one does not anticipate the trajectory, therefore in sharp curves this solution tend to adapt the direction too late and let the vehicle goes to the outside of the curve (overshooting).

3. Prerequisites

3.1. Positionning system

On the one hand, one of the main requirement of the demonstration is to have accurate data for the position and the heading of the vehicle. On the other hand, the challenge is to control the vehicle without sensing the environment. Simply speaking, all sensors are switched off. Only a global positioning system coupled with wireless communication are usable. This dramatically increase the sampling time of the system. Hence, to achieve the objective we use a Real-Time Kinematic (RTK) positionning system, which allows an error about 20cm in open-ground.

3.2. Controllable car

The presented solution requires a car whose steering wheel angle can be controlled, as well as its acceleration and braking commands.

4. Vehicle control

4.1. Wheel angle control

In order to get the car to follow the path, we have to compute the wheel angle according to the current position and orientation of the car.

The principle of the algorithm used to compute the direction is an improved version of the pure pursuit algorithm, designed to consider the period of the positioning system in order to compute the position of the followed point.

A reference track is defined by using a previous recording of the vehicle trajectory. The reference track is a set of points built by one tour of the vehicle driven by a human. We use the spline interpolation to match the points. Besides, Two limits of the track, i.e. left side and right side, are considered. All vehicles move in the same direction. The curvature of the opposite side of the turning movement is divided by 1.2. The target point is then in the middle of both sides. A target point is defined on this track according to the current position,

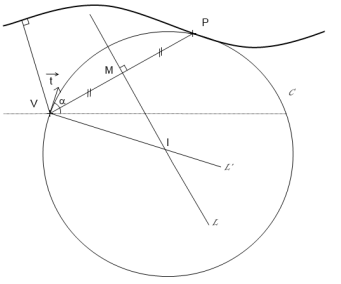

Fig. 2: Scheme of the wheel angle computation algorithm

the current speed of the vehicle and the period of the positioning system. Knowing the current heading of the vehicle, the steering wheel angle required to make the vehicle reach this point is computed using the "bicycle" car model. The details of the algorithm are presented in Algorithm 1.

Algorithm 1 Wheel angle computation algorithm

Compute the position V of the middle of the vehicle rear axle and the direction vector $\overrightarrow{t}$ of the car
Project the position on the path and get the curvilign abscissa s of this projected position
Get the point P whose curvilign abscissa is $s + m$ where $m = s_0 + \tau * v$
Compute the radius R of the unique circle C which contains V, P and whose tangent in P is directed by $\overrightarrow{t}$
Compute the angle of the front wheels using the formula $\delta = atan(\frac{E}{R})$ where E is the distance between the two axles of the car

Lemma 1. *According to this algorithm, the angle of the front wheels δ is equal to*

$$\delta = atan(\frac{2E((y_p-y_v)cos(\alpha)-(x_p-x_v))sin(\alpha)}{(x_p-x_v)^2+(x_p-x_v)^2})$$

Proof 1. *From the definition of C, V and P belong to C, so the center of C is equidistant to V and P, thus it belongs to the line $\mathcal{L} = (MI)$ segment bissector of $[VP]$.*

Moreover, the line $\mathcal{D}$ directed by $\overrightarrow{t}$ passing by V is tangent to C, so the line $\mathcal{L}' = (VI)$ perpendicular to $\mathcal{D}$ passing by V contains the center of the circle C.

$\mathcal{L}$ is perpendicular to $[VP]$ so it is directed by the normal of $\overrightarrow{VP}$, $n_{\overrightarrow{VP}}\begin{pmatrix}x_c - x_v\\ y_c - y_v\end{pmatrix}$. Hence the parametric equation of $\mathcal{L}$ is

$$\begin{cases} t \times (y_p - yv) + \dfrac{x_p + x_v}{2} & (2)\\ t \times (x_v - xp) + \dfrac{y_p + y_v}{2} & (3)\end{cases}$$

.

$\mathcal{L}'$ is directed by the normal of $\vec{t}$ named $\vec{n}$. Hence, its parametric equation is

$$\begin{cases} t \times cos(\alpha) + x_v & (4)\\ t \times sin(\alpha) + y_v & (5)\end{cases}$$

.

Knowing the parametric equations of $\mathcal{L}$ and $\mathcal{L}'$, their intersection point can be computed as the solution of the system

$$\begin{cases} t \times cos(\alpha) + x_v = t' \times (y_p - yv) + \dfrac{x_p + x_v}{2} & (6)\\ t \times sin(\alpha) + y_v = t' \times (x_v - xp) + \dfrac{y_p + y_v}{2} & (7)\end{cases}$$

.

Thus, the radius of the circle can be expressed as

$$\begin{aligned} R &= \sqrt{(t'(y_c - y_v) + \frac{x_p - x_v}{2})^2 + (t'(x_v - x_c) + \frac{y_c - y_v}{2})^2} \\ &= \frac{(x_p - x_v)^2 + (x_p - x_v)^2}{2((y_p - y_v)cos(\alpha) - (x_p - x_v))sin(\alpha)} \end{aligned} \quad (8)$$

.

Finally, according to the relation $\delta = atan(\frac{E}{R})$ the wheel angle can be expressed as $\delta = atan(\frac{2E((y_p-y_v)cos(\alpha)-(x_p-x_v))sin(\alpha)}{(x_p-x_v)^2+(x_p-x_v)^2})$.

To improve the accuracy of the pure-pursuit, the position of the followed point has to depend on the current speed of the car, and on the estimated reaction delay of the system. Indeed, lets suppose the following conditions:

- The positionning system of the car gives an updated position every τ seconds
- The curvilign distance between the projected position of the car and the followed point is l
- The curvilign speed of the projected position of the car is $v = \frac{ds}{dt}$ with s the curvilign absissa of the projection of the car on the track (the curvilign abscissa is assumed continuous and differentiable).

At time t, the car computes the steering wheel angle according to its known position. The next steering wheel angle will be computed at time $t + \tau$. Then, there is two possible situations: at $t + \tau$ the car is further than the previously followed point, or not, i.e. $v \times \tau > l$ or $v \times \tau < l$.

The objective of the computed steering wheel angle is to reach the followed point, thus if $v \times \tau > l$, it means that the car has continued on the track with a steering wheel angle which is not relevant anymore. It will then introduce oscillations in the behavior of the car.

To avoid these oscillations, the following condition must be respected: $v \times \tau < l$. Therefore l is chosen so $l > v \times \tau$. However, if l is too important the pure pursuit algorithm tends to take shortcuts in curves. Thus, if we want to keep l small in curves, it is needed to reduce v in order to satisfy the equation $l > v \times \tau$.

4.2. Improvement by applying a coefficient on the computed radius

The pure-pursuit algorithm is subject to the "cutting corners" effect, i.e. when the look-ahead point is set far, the car will tend to cut the trajectory in sharp curves. The usual solution to avoid this issue is to reduce the look-ahead distance, but this solution usually increase the oscillations of the car. In this section, an empirical improvement to the pure-pursuit is presented in order to increase path-following performance without reducing the look-ahead distance, and without increasing oscillations.

In the previous part, the wheel angle was defined by $\delta = atan(\frac{E}{R})$. In this part, a new formula is defined empirically by $\delta = atan(k \times \frac{E}{R})$ where k is a coefficient. The coefficient k is a constant without dimension whose purpose it to change the curvature of the arc drawn by the pure pursuit. Values above 1 will increase the curvature while values below 1 will decrease it. Therefore it can be used to reduce the surface S between the "pure-pursuit curve" and the real trajectory. Figure 3 illustrates the relation between k and S.

It has been verified either in simulation or using a real vehicle that choosing a proper value for k increases the track-following performance. In our tests k has been set to $k = 1 - \alpha S$ where α is a coefficient arbitrarily defined. Our simulation had driven us to set a value of 0.02 for α. Moreover, the resulting k value has been limited to $[0.7, 1.3]$. Comparison with, or without, k is made in figure 4 (simulation made with a simulated error on the position and the heading, in order to reproduce the behavior of a GPS).

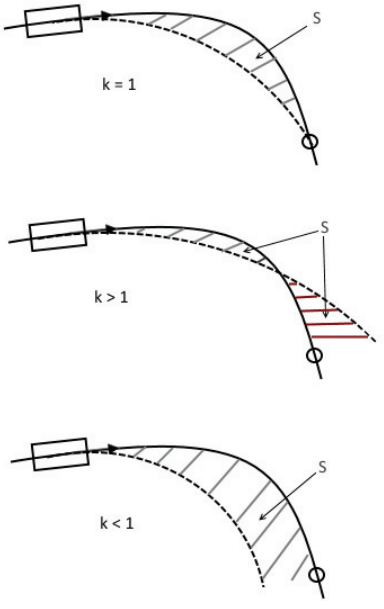

Fig. 3: Influence of coefficient k on S

4.3. Acceleration control

As said before, the major drawback of the pure pursuit algorithm is that, when the speed increases, the car tends to cut the corners of the trajectory. To avoid this problem we have to reduce the speed of the car when there is an important change in the trajectory curvature.

For this purpose, we create virtual border lines around the path that will act as obstacles for the car. Then, we compute the acceleration of the car in a way its speed must be null when the distance with the virtual borders is null.

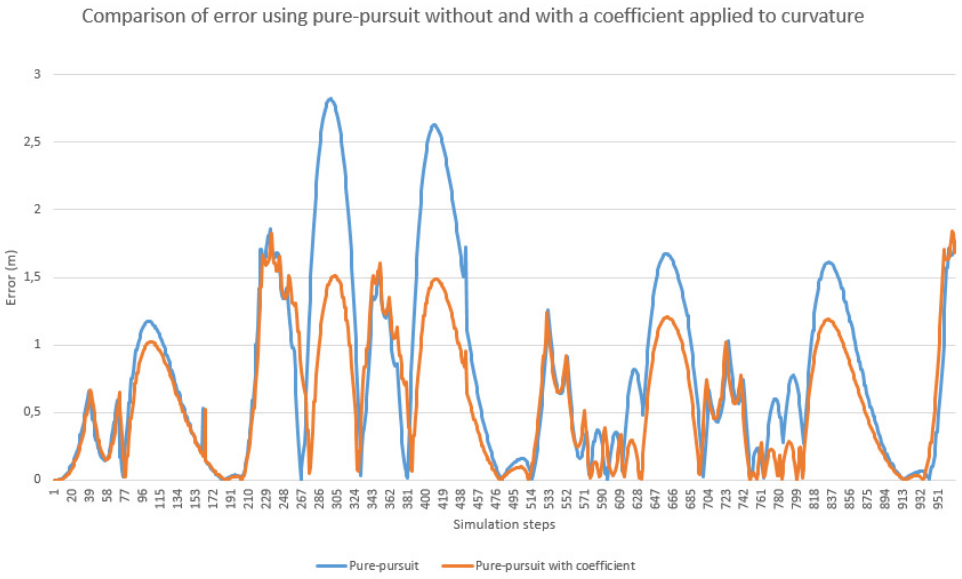

Fig. 4: Comparison of error between the pure-pursuit and the improved pure-pursuit

The computed acceleration is then computed using the following formula[1]:

$$a = \frac{b_f \tau - 2v_f - 2b_f \sqrt{\frac{b_f b_l \tau^2 + 4 b_l v_f \tau - 8 b_l s}{4 b_f b_l}}}{2\tau} \tag{9}$$

With :

- b_f the desired deceleration of the vehicle
- b_l the maximum braking of the vehicle
- v_f the current speed of the vehicle
- τ the maximum reaction time of the system
- s the distance with the border

The main advantage of this formula is that it allow us to set a "reaction time" of the system (the τ parameter). The reaction time here is determined by the sum of the positionning system period (or latency) and the latency of the acceleration/braking command system.

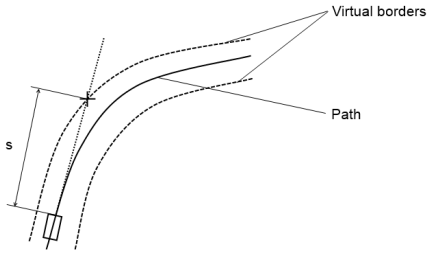

Fig. 5: Scheme of the acceleration computation algorithm

5. Tests and results

Before testing the control approach by using real cars, we did many simulations for validation. We draw the reader attention to the fact that the speed of the vehicle increase the control error. This is also true for human drivers. Indeed, by means of Oculus Rift and a steering wheel many drivers have tried to control the vehicles with different speed. The comfortable vehicle speed in the turning movement of the circuit is bellow $3m.s^{-1}$ whereas in the straight lines the vehicle can reach $10m.s^{-1}$. These values were also confirmed by computing lateral and longitudinal accelerations as well as jerk. In order to consider the limit of the proposed approach, the presented simulation doesn't respect the speed limitations. Instead it considers tuning movements at $10m.s^{-1}$. This allows us to easily compare and distinguish the approaches. This also allows automatically stop the vehicles if the conditions are not fulfilled. In the following

we have considered that an error more than $1m$ is not acceptable. By considering the speed limitations as discussed earlier, the errors shown in Figure 4 were dramatically decreased.

5.1. Simulation

The first step of our development process was to make simulations of our algorithm. Thus, we have developped a simulator in Java dedicated to the test of this control of direction.

We used an agent-based simulation where a reactive agent (the control algorithm) was controlling a "body" (a virtual representation of the vehicle and its sensors) in a virtual environment. This seperation between the intelligence and the virtual body of the agent allowed us to introduce easily errors (random noise) in the information used by the control algorithm (i.e. innacuracy or latency of the positionning system), and in the effectiveness of the command (i.e. error and latency in the execution of the direction control). Therefore we were able to challenge the algorithm in degraded conditions.

In the simulator we made the virtual vehicle follow different kind of tracks. Given that the algorithm purpose is to reproduce a real path followed by a vehicle, the following constraints where set on the tracks :

- the track has to be continuous (no gap)
- the track has to be almost smooth (no sharp angles or corners)

Our objective when making these simulations was to implement the control algorithm on a Renault Scenic III. Therefore we have set the following parameters for our simulation:

- Dimensions according to figure 6
- Acceleration limited to $2m.s^{-2}$
- Deceleration limited to $-7m.s^{-2}$

We then made tests, measured the average error (average distance between the vehicle and the track) and we defined limits for the position of the vehicle: any part of the vehicle must not be further than 2.5 meters from the center of the track, otherwise the simulation would be considered as failed. An average error is also computed: at each simulation step, the absolute distance between the center of the car and the track is computed and the mean of this value is then computed. The test results presented here were obtained on an eight-shaped circuit. They are detailed in 1.

Position error	0.1 m	0.1 m	0.1 m	0.1 m	0.1 m	1.0 m	1.0 m	1.0 m	2.0 m
Heading error	5	15	5	5	15	5	15	5	5
Wheel angle latency	200 ms	200 ms	500 ms	200 ms	500 ms	200 ms	200 ms	200 ms	200 ms
Wheel angle error	1	1	1	10	10	1	1	10	1
Failure	No	No	No	No	Yes	No	Yes	No	Yes
Average error	0.42 m	0.45 m	0.43 m	0.42 m	0.50 m	0.54 m	0.54 m	0.52 m	0.80 m

Table 1: Simulation results

With our simulations, we observed that a precise positionning solution was required: an error on the position above 1 m is too important for a precise track following. Therefore a standard GPS positionning system is not usable, a Wide Area Augmentation System (WAAS) could be usable if the error on the control is low, but currently an RTK positionning system is the more reliable solution. Moreover we have noticed that the adaptation of the acceleration in the curves of the track were able to balance the delay of application of the command.

5.2. Experimentations

In order to prove the efficiency and accuracy of our solution, experimentations have been made using real vehicles. The model used was a Renault Scenic III (see 6 for relevant specifications of the vehicle). This vehicle was robotized

by FAAR Industry so the acceleration, braking pedal, and steering wheel angle where controlable programmatically with messages sent on a CAN bus.

Length	4344 mm
Width	1845 mm
Wheelbase	2703 mm
Turning radius	5645 mm
Front track	1546 mm
Rear track	1547 mm

Fig. 6: Dimensions of the Renault Scenic III used for the tests

The positionning system used was a GNSS RTK ProFlex 500 in order to have a precise position. Our experimentations have shown an error less than 30 cm for position and 10 degrees for heading most of the time, but for the sake of safety, we managed to make the vehicle brake when the RTK rover had positionning issues or communication issues with the RTK base. Therefore the vehicle was only allowed to move if the mode was "RTK fixed" or "RTK float" with a maximal communication delay of 4 seconds in this second case.

Our experimentation protocol was then close to the one used in the simulations: the goal was to follow a saved track on an eight-shaped circuit. The test track was delimited with lane separators on the left and on the right side of the track. The lane width was fixed to 6 m, so there were about 2.1 meters of space on each side of the vehicle when it was on the center of lane, thus forbidding big errors.

The first step of the experimentation was to record the points of the track by making manually a first passage on the track. The second step was to let the car try to follow the registered car without any human intervention.

Only the RTK system was used as sensor to control the vehicle, with a sampling time of 500ms. Due to the reduced speed, errors were slightly smaller than the ones depicted in figure 4. Moreover all vehicles behave similarly backing to the same track at each lap. This is due to the speed control and also to the fact that the initial track is fitted.

6. Conclusion

This article presents a solution to overcome the drawbacks of the pure-pursuit algorithm by controlling simultaneously the steering wheel angle and the acceleration of the car.

The key points are:

- the definition of a point to follow according to the speed and the period of the positionning system
- the computation of the steering wheel angle
- the adaptation of the speed according to the curvature of the track in order to improve the path tracking performance

In order to validate the proposed solution, tests have been made in simulation, and using a real controllable car. Moreover this solution has already been demonstrated in the Intelligent Transportation System World Congress 2015 in Bordeaux as a requirement of the "speed synchronization at intersections using V2X communication" demonstration.

References

1. X. Hao, A. Abbas-Turki, F. Perronnet, and R. Bouyekhf, "V2i-based velocity synchronization at intersection."
2. J. M. Snider, "Automatic steering methods for autonomous automobile path tracking," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.
3. M. Samuel, M. Hussein, and M. B. Mohamad, "A review of some pure-pursuit based path tracking techniques for control of autonomous vehicle," *International Journal of Computer Applications*, 2016.
4. R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," DTIC Document, Tech. Rep., 1992.
5. S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
6. J. Wit, C. D. Crane, and D. Armstrong, "Autonomous ground vehicle path tracking," *Journal of Robotic Systems*, vol. 21, no. 8, pp. 439–449, 2004.
7. Y. H. Cho and J. Kim, "Design of optimal four-wheel steering system," *Vehicle System Dynamics*, vol. 24, no. 9, pp. 661–682, 1995.