Proceedings of the 2006 IEEE International Conference on Robotics and Automation
Orlando, Florida - May 2006

Trajectory deformation applied to kinodynamic motion planning for a realistic car model

Fabrice Boyer
LAAS-CNRS / PSA Peugeot Citroën
18, allée des fauvelles
92200 La Garenne-Colombes France
E-mail : fabrice.boyer@laas.fr

Florent Lamiraux
LAAS-CNRS
7 avenue du colonel Roche
31077 Toulouse Cedex 4 France
E-mail : florent@laas.fr

***Abstract* – Providing a safe behavior of a vehicle in every situation is a critical mission in the design of a modern car. To measure this ability, the standard ISO double lane change test was designed by consumer unions. The detailed simulation of such an active safety issue would contribute to shorten the design cycle time of a vehicle by reducing the number of real world tests. In this paper we will discuss the use of a trajectory deformation algorithm to determine a possible motion for a realistic dynamic car model going through such a test, and the way to determine the maximum passage speed. The main idea of this method is to iteratively deform the inputs of the system to progressively reduce the number of collisions while respecting the dynamic constraints. This deformation is calculated for each iteration by taking into account a locally linearized dynamic model of the system. Consequently, this algorithm can anticipate correctly the inputs needed to solve the future collisions.**

***Index Terms* – Trajectory deformation, motion planning, realistic dynamic car model, ISO double lane change test, motion optimization.**

I. Introduction

Ensuring a safe behavior for a vehicle in any situation is a critical mission in the design of modern cars. To evaluate this active safety issue, tests need to be designed and performed. The most classical ones consist in emergency avoidance tests performed at different speeds and lane configurations. Since these physical tests are difficult and time consuming, their simulation would be helpful to car designers. This article proposes to transfer and adapt techniques coming from the domain of robotics to the domain of vehicle design.

A. The ISO double lane change test

This test, which is the central test case of our study, was originally designed by consumer unions and standardized by ISO later on (ISO 3888-1). The test consists in going through a first lane at the highest possible speed, then reaching a second shifted lane before coming back to the original trajectory in the last corridor (Fig 1). The objective is to simulate an emergency avoidance maneuver so as to evaluate the stability of the vehicle in such extreme conditions. The whole test is performed by steering only, i.e. no motor or braking effort is applied once entered in the first lane (Fig 2). This test is considered as a very difficult one because reaching the highest speed requires both a strong driving ability and a fine understanding of the physics of the vehicle being tested. Consequently, the simulation and the discovery of such a subtle dynamic motion is challenging for any algorithm.

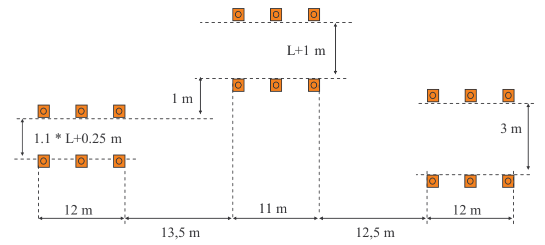

Fig 1 Dimensions of the standard ISO double lane change test
(L is the width of the vehicle in meters)

B. The classical motion simulation techniques in the industry

Nowadays, the most frequently used methods to simulate those avoidance maneuvers are the open loop simulation with a measured steering angle feeding the vehicle model, and the closed loop simulation where the model is tracking a predefined trajectory presumed optimal. These generic approaches can deliver interesting clues about the abilities of the vehicle, but do not offer any guarantee that the vehicle really passes the test without any collision. Other techniques are available, like the optimization of a parameterized steering profile. But all these methods are rather far away from the real world tests, where the test driver has to find a trajectory for each particular car configuration in order to clear the test as fast as possible. These observations led to the intuition that motion planning algorithms, which were precisely designed to solve such problems, could help the car manufacturer to improve its simulation methodologies.

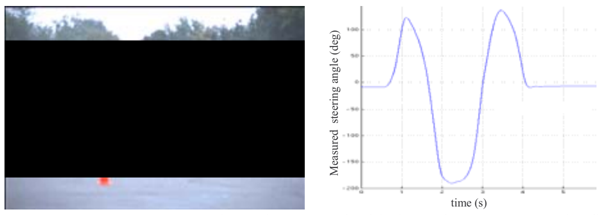

Fig 2 A Peugeot 307 going through an ISO double lane change test at 70km/h and the corresponding steering angle (deg)

The research domain of kinodynamic motion planning is very challenging here because the industrial car model we used is a black box and contains numerous state variables. The analytical formulas defining the dynamic model of the vehicle are thus not accessible to us.

II. The existing kinodynamic motion planning algorithms

A. *The general kinodynamic model*

The system considered at a time $s \in [0, S]$ is reduced to its state space representation, $\mathbf{q}(s)$ where vector $\mathbf{q}(\mathrm{s}) = (\mathrm{q}_1(\mathrm{s}), \ldots, \mathrm{q}_\mathrm{n}(\mathrm{s}))$ contains the n continuous scalar states of the system. We use a dynamic model, therefore $\mathbf{q}(s)$ not only includes classical state variables like the position x, y of the vehicle, but it also contains their derivates $\dot{x}, \dot{y}$ to take into account dynamic phenomenon like drift or inertia. The evolution of the whole dynamic system is modeled with the classical general equation $\dot{\mathbf{q}}(s) = \mathbf{f}(\mathbf{q}(s), \mathbf{u}(s))$, where $\mathbf{u}(\mathrm{s}) = (\mathrm{u}_1(\mathrm{s}), \ldots, \mathrm{u}_\mathrm{k}(\mathrm{s}))$ is a vector containing the k scalar inputs of the system. From now on, notation $\dot{\mathbf{q}}$ denotes the derivative of $\mathbf{q}$ with respect to time s.

B. *Exploration algorithms*

Our problem can be posed as a classical kinodynamic motion planning problem. However, for the dynamic system we are considering, no local method is available to connect two given states. Therefore the number of available algorithms is drastically reduced. The algorithms we are presenting in this section are all based on a guided exploration of the state space. Their principle is to dynamically construct a tree containing states of the system.

The root of the tree is the initial state; from there l inputs are applied to $\dot{\mathbf{q}}(s) = \mathbf{f}(\mathbf{q}(s), \mathbf{u}(s))$ during a time Δt. These l short simulations generate l new states. These states are added to the tree if they are not in collision and respect a filtering criterion that avoids a combinatorial explosion. Afterward a node of the tree is selected, and other inputs are tested. The loop goes on until the goal region is reached by the tree (Fig 3). The solution is read by going backward through the tree from the end to the origin.

These algorithms vary by the selection method of the current node, the manner of selecting the applied inputs, and the criterion accepting or not the nodes in the tree. For instance, FDP algorithm [1] chooses a leaf thanks to a cost calculated for every node, then tries a fixed set of inputs and filters the generated states with a grid allowing only one node per cell. The RRT algorithm [2] randomly chooses a state and selects the nearest node of the actual tree (using a certain metric). From there, it applies the input that makes the new node as close as possible to the randomly chosen node. Another example is KDP algorithm [3] which randomly chooses a leaf and applies a random input during a random time Δt. Note that numerous other variants exist for this category of algorithms.

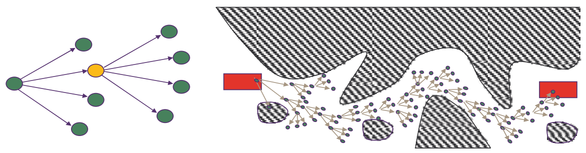
Fig 3 Exploration algorithms can find solutions to dynamic motion planning problems only in low dimensions spaces and without much accuracy

C. *Our solution*

The main drawback of these techniques is their intrinsic lack of accuracy. This problem is mainly caused by the large time step Δt and the poor number of tested inputs needed to restrict the number of computed nodes. A trade-off has to be established between computing time and accuracy of the exploration. The problem encountered by these algorithms is as hard as trying to find a long and narrow corridor connecting two large empty regions of state space [4]. Meanwhile, in our avoidance tests, the narrowness of the corridor is mainly due to the dynamics of the system. Improvements were brought to these algorithms to deal with dynamic systems (Cf [5], [3]). We have implemented these improvements but we did not manage to significantly improve the determination of the maximum speed allowing a car to successfully pass the ISO double lane change test (from now on, we will refer to this speed as the maximum passage speed).

Nevertheless, when trying to compute low speed motions through such tests, exploration algorithms are very efficient because the state space is less constrained by the dynamics (and the corridor of solutions is much wider). The idea presented here is to first use one of these algorithms at a low speed so as to obtain an initial possible trajectory that clears the test. Then, we increase the initial speed of the trajectory while using the same steering angle profile. This simulation generates several collisions (Fig 9). We solve these collisions using the trajectory optimization algorithm presented in III. This algorithm is a local optimization algorithm the state of which is a trajectory. The input of the algorithm is thus an initial trajectory. Trajectory optimization techniques already exist (like [6] or [7]), but few address the general dynamic case we are interested in.

III. Theory of trajectory deformation

The algorithm we are presenting here is based on the work presented in [8]. The notations we recall here are taken from this article. We have extended this algorithm to address the particular issues of our problem (dynamic model, no final constraints…)

A. *The inputs*

The inputs of the algorithm are:

- The obstacles and vehicle geometry definition
- The dynamic model of the car (1)
- An initial state configuration.
- An initial guess for the input $\mathbf{u}(\mathrm{s})$ provided by II

B. *Principles of the trajectory deformation*

The basic idea of this method is to iteratively deform the trajectory **q**(s) by perturbing the input **u**(s). Therefore we have to work on a set of trajectories $\mathbf{q}(s,\tau)$ similar to the one presented in Fig 4. This set is indexed by parameter $\tau \in [0,+\infty[$ that is increased between two iterations (as described in [8]). Notice that $\mathbf{u}(s,\tau)$ is the input corresponding to the deformed trajectory $\mathbf{q}(s,\tau)$.

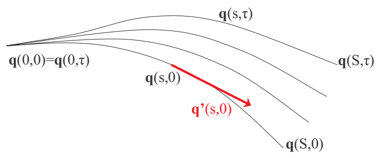

Fig 4 Each value of the variable τ defines a different trajectory

The dynamic system of II.A can now be rewritten as:

$$\frac{\partial \mathbf{q}}{\partial s}(s,\tau) = \dot{\mathbf{q}}(s,\tau) = \mathbf{f}(\mathbf{q}(s,\tau),\mathbf{u}(s,\tau)) \quad (1)$$

If we derivate this expression with respect to τ, we obtain:

$$\frac{\partial \dot{\mathbf{q}}}{\partial \tau} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}}(\mathbf{q},\mathbf{u}).\frac{\partial \mathbf{q}}{\partial \tau}(s,\tau) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{q},\mathbf{u}).\frac{\partial \mathbf{u}}{\partial \tau}(s,\tau)$$

Let us define **A**(s,τ) as the following $n \times n$ matrices:

$$\mathbf{A}(s,\tau) = \frac{\partial \mathbf{f}}{\partial \mathbf{q}}(\mathbf{q}(s,\tau),\mathbf{u}(s,\tau)) \quad (2)$$

and **B**(s,τ) the $n \times k$ matrices:

$$\mathbf{B}(s,\tau) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{q}(s,\tau),\mathbf{u}(s,\tau)) \quad (3)$$

We call respectively *input perturbations* and *directions of deformation* the following vector valued functions:

$$[0,S]\times[0,+\infty[\to \mathbf{R}^k \qquad\qquad [0,S]\times[0,+\infty[\to \mathbf{R}^n$$
$$\mathbf{v}(s,\tau) = \frac{\partial \mathbf{u}}{\partial \tau}(s,\tau) \quad \text{and} \quad \boldsymbol{\eta}(s,\tau) = \frac{\partial \mathbf{q}}{\partial \tau}(s,\tau)$$

With this notation, we obtain the time dependent differential equation (4) that is the linearized system about the current trajectory $\mathbf{q}(s,\tau)$:

$$\dot{\boldsymbol{\eta}}(s,\tau) = \mathbf{A}(s,\tau).\boldsymbol{\eta}(s,\tau) + \mathbf{B}(s,\tau).\mathbf{v}(s,\tau) \quad (4)$$

To be able to integrate this equation, the initial condition $\boldsymbol{\eta}(0,\tau) = \mathbf{0}$ is added (i.e. the initial state must not move). This equation allows the calculus of the direction $\boldsymbol{\eta}(s)$ in which the trajectory will move when the perturbation $\mathbf{v}(s)$ is applied (Fig 5).

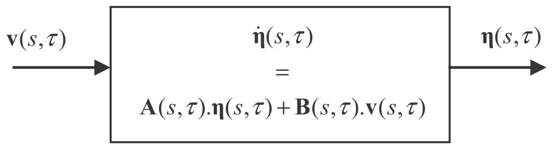

Fig 5 The dynamic system giving the direction of deformation induced by a perturbation

C. *Decomposition of the perturbation*

The perturbation belongs to the C^∞ infinite-dimensional vector space. To approximate this vector space, we choose **v**(s) in a finite-dimensional subspace generated by the basis $\{\mathbf{e_i}(s)\}_{i\in[1,p]}$ of *p* linearly independent functions (*p*>*n*):

$$\mathbf{v}(s,\tau) = \sum_{i=1}^{p} \lambda_i(\tau)\mathbf{e_i}(s) \quad (5)$$

Consequently, the corresponding direction of deformation $\boldsymbol{\eta}(s)$ can be decomposed into elementary directions of deformations $\mathbf{E_i}(s)$:

$$\boldsymbol{\eta}(s,\tau) = \sum_{i=1}^{p} \lambda_i(\tau)\mathbf{E_i}(s,\tau) \quad (6)$$

The elementary directions of deformations $\mathbf{E_i}(s,\tau)$ can be computed for every perturbation $\mathbf{e_i}(s)$ by integrating (4). These $\mathbf{E_i}(s,\tau)$ represent the deformation corresponding to the application of the elementary perturbation $\mathbf{e_i}(s)$ to the linearized system.

D. *The potential function*

A local potential $U(\mathbf{q}(s,\tau))$ is introduced for every state configuration. It is designed to be strong near obstacles and zero when far away from them. To characterize the proximity of the trajectory to the obstacles, we introduce a global potential function $V(\tau)$ that sums up the elementary potentials $U(\mathbf{q}(s,\tau))$ along the trajectory:

$$V(\tau) = \int_0^S U(\mathbf{q}(s,\tau))ds$$

The objective is to get this global potential function diminishing, therefore we need:

$$\frac{dV}{d\tau}(\tau) = \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s,\tau)).\frac{\partial \mathbf{q}}{\partial \tau}(s,\tau).ds < 0$$

$$\text{i.e. } \frac{dV}{d\tau}(\tau) = \sum_{i=1}^{p} \lambda_i(\tau) \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s,\tau)).\mathbf{E_i}(s,\tau).ds < 0$$

This can be ensured by choosing:

$$\lambda_i(\tau) = -\int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s,\tau)).\mathbf{E_i}(s,\tau).ds \quad (7)$$

The $n \times 1$ gradient $\frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s,\tau))$ can be assimilated to a force in the configuration space (Fig 6), this force pushes the trajectory away from obstacles. Directions of deformation $\boldsymbol{\eta}(s,\tau)$ are computed thanks to (7) and (6) . These directions are different from the directions of the forces because they take into account the dynamic behavior of the system. This allows our extension of the trajectory optimization algorithm to anticipate the inertia and the drift of the system to solve the punctual collisions represented by the forces.

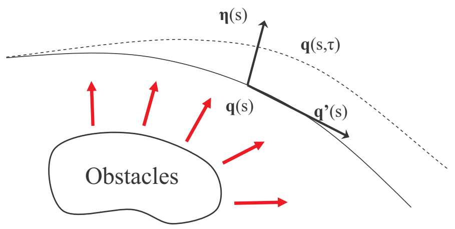

Fig 6 The iterative deformation process, where the trajectory goes away from the obstacles under the action of forces while respecting the dynamic constraints

E. *Perturbing the inputs*

The maximum intensity K of the forces can be as strong as we want since we are only interested in their directions. Therefore, the parameters $\Delta\tau$ and $\boldsymbol{\eta}_{\max}$ are introduced to limit the intensity of the resulting trajectory perturbation. Parameter $\eta_{\max}$ is experimentally chosen to ensure that the trajectory won't move too much between two iterations and will respect the linearization computed in (4). We impose:

$$\begin{cases} \Delta\tau = \dfrac{\boldsymbol{\eta}_{\max}}{\|\boldsymbol{\eta}\|_\infty} \; if \; \|\boldsymbol{\eta}\|_\infty \geq \boldsymbol{\eta}_{\max} \\ \Delta\tau = 1 \quad otherwise \end{cases}$$

To deform the input, we simply add to the current input the deformation we have calculated with (5) and (7) and limit the intensity of the deformation with the global coefficient $\Delta\tau$:

$$\mathbf{u}(s,\tau) = \mathbf{u}(s,\tau) + \Delta\tau.\mathbf{v}(s,\tau) \tag{8}$$

The iteration is now over, we can increment the index τ of the trajectory set: $\tau \leftarrow \tau + \Delta\tau$ and loop again until all collisions are solved.

F. *Summary of the algorithm*

The algorithm consists in iterating over the following loop until all the collisions are eliminated:

1. Compute **q**(s) by integrating (1) with III.A
2. Compute **A**(s) and **B**(s) by finite differences of (2),(3)
3. Compute every $\mathbf{E_i}(s)$ by integrating (4)
4. Compute the gradient of U(s) by detecting collisions
5. Compute each λ_i by integrating (7)
6. Compute **v**(s) using (5)
7. Deform the inputs using (8)

IV. Implementation

A. *The industrial car model*

The dynamic car model we used is named SimulinkCar and is implemented on the Matlab/Simulink™ platform. It has 24 dimensions (n = 24) and can be parameterized by a configuration file generated by PSA Peugeot Citroën engineers to adapt the model to the car being simulated. This file sets up numerous variables like mass, inertia, tire model parameters, or the response cartography of springs, dampers… This model is precise and fast enough to meet our requirements. The ISO double lane change test is performed by steering only, therefore our input **u**(s) is reduced to the one-dimensional steering wheel angle value (k = 1).

B. *Numerical approximations*

The initial trajectory is sampled using approximately 100 points for a 4 seconds test run (much more than what was possible in II). As a consequence, it is an equivalent discrete version of the algorithm presented in III that we are currently using (Cf [8]). The Simulink modeling used by our algorithm is a black box, i.e. no analytical expression of (1) is available, and the calculus of the **A** and **B** matrices has to be done by finite differences. This process is extremely time-consuming and not very precise; this point is one the main drawback of the implementation. The $\{\mathbf{e_i}(s)\}_{i\in[1,p]}$ functions are chosen as a classical Fourier basis decomposition. Since, we want our one-dimensional perturbation to be equal to zero in $s = 0$, we only need a weighted sum of sine functions:

$$\forall i \in [1,p], \forall s \in [0,S] \quad e_i(s) = \sin\left(\frac{i.\pi.s}{S}\right)$$

This Fourier decomposition can create unwished border effects in several zones of the trajectory (typically, beginning and end). These border effects may result in perturbations without real physical meaning. From now on, notation x,y denotes the physical position of the center of gravity of the vehicle, the ISO double lane change test is oriented along the increasing x axis values. Fig 7 presents examples of the three first steering perturbations e_1, e_2 ,e_3 and the three resulting directions of deformation $\mathbf{E_1}$, $\mathbf{E_2}$, $\mathbf{E_3}$ (projected along the transversal y direction). The perturbation e_1 corresponding to a progressive steering angle on the left and a return to the origin results in a trajectory deformation $\mathbf{E_1}$ on the left (i.e. the car turns left), what is perfectly coherent. We plan to test other function bases in the future.

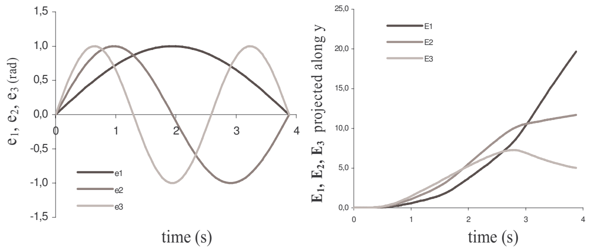

Fig 7 Projection along y of the 24x1 vectors $\mathbf{E_1}$, $\mathbf{E_2}$ and $\mathbf{E_3}$ corresponding to the 1x1 perturbations e_1, e_2 and e_3

C. *Gradient calculus and collision detection algorithm*

We used the PQP collision detection package from the University of North Carolina [9] to detect the collision of our car geometry with the defined obstacles. In case of collision, we chose to apply forces only in the y direction: the bottom obstacles (right of the car) are pushing upward while the top obstacles are pushing downward (Fig 8). We apply directly:

$$\frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s,\tau)) = \left(0,\ \pm K,\ \underbrace{0,\cdots,0}_{22\ times}\right)$$

This simple gradient works well in our case but is not optimal and probably causes unnecessary iterations. More complex gradients will be tested, including a rotational moment depending on the impact point of the collision on the car.

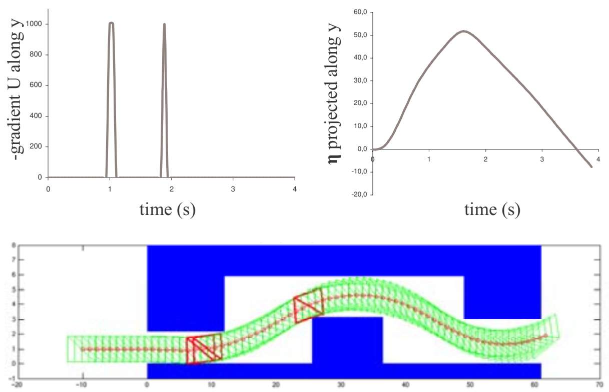

Fig 8 The direction of deformation **η** is mainly along the y axis due to the simple y oriented gradient we chose here

D. *Determination of the maximum passage speed*

When a solution is found, the vehicle initial speed is increased and the same input is applied. We observe on Fig 9 that new collisions occur and the trajectory has to be deformed to solve these collisions.

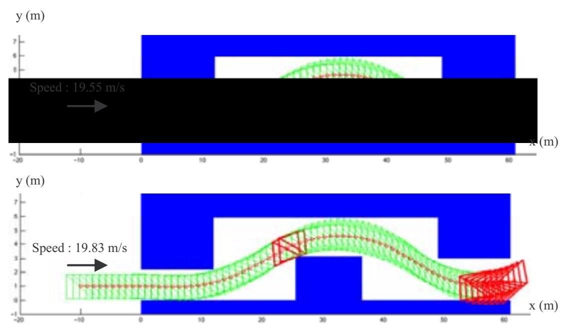
[redacted]

Fig 9 Collisions generated by changing the speed of the vehicle while feeding the same input

After a certain amount of unsuccessful iterations, we stop and consider the last good speed as the maximum passage speed. The $\eta_{\max}$ parameter need to be small enough to ensure that the iterative process won't oscillate because of too important perturbations between two iterations. In our example, we stopped after 100 unsuccessful iterations, with $\eta_{\max} = 0.1$, and a speed increment of 1km/h.

V. Experimental Results

The main objective here is to be able to robustly predict the influence of a parameter of the model over the maximum passage speed of the test. The correlation with the real maximum passage speed of the real vehicle is not important here. To test this ability, we performed several tests consisting in simple parametric modifications of the model. These modifications are chosen to improve the maximum passage speed of the car. Such influent parameters can be: a lighter weight, a better yaw inertia, or parameters of the tire model.

A. *Lateral behavior of a tire*

When a tire is drifting i.e. when its real orientation is different from the one of its speed vector. The tire generates a lateral force F_y that allows the car to turn (Fig 10). This force is modeled by several parameters in the Pacejka tire model: the cornering stiffness is the slope at the origin, and the asymptote of the curve can be considered as an image of the maximal adhesion of the tire. These values depend on the vertical force F_z applied to the tire.

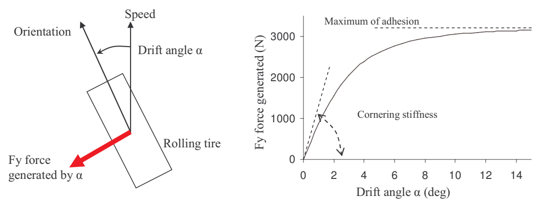

Fig 10 The force generated by the drifting of the tire allows the car to turn

B. *Elementary tests*

We performed several simple improvements on the cornering stiffness and the maximum of adhesion (Fig 11) to test their influence on the maximum passage speed of the test.

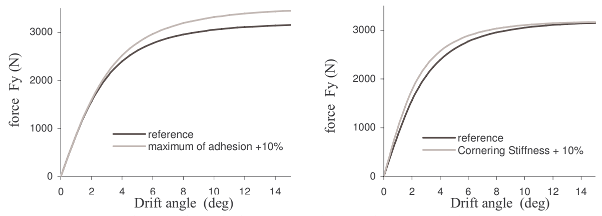

Fig 11 The modified characteristics of the tire, with F_z=3000N

The results of these tests are presented on Table 1. The algorithm seems to be predictive and robust when trying to determine this speed. Many other parameters can be tuned, so we can observe their impact on the maximum passage speed of the ISO double lane change test.

Vehicle	ISO double lane change maximum passage speed
Reference	19.6 m/s
Maximum of adhesion * 1.1	20.1 m/s
Cornering stiffness * 1.1	20.1 m/s
Maximum of adhesion * 1.1 + Cornering stiffness * 1.1	20.8 m/s

Table 1 Impact of various improvements of the tire behavior on the maximum passage speed of the ISO lane change.

An example of a real steering angle and the result of a simulation are presented on Fig 12. The profiles of the two curves are very close with three main peaks at ± 2 *rad* corresponding to the minimum steering actions needed to pass such a test. Nevertheless several differences exist between the two curves. For instance the small oscillation applied at the beginning of the simulated input (doted circle) allows the car to pass the test more easily. This maneuver is very difficult to realize in real tests because of the narrowness of the first corridor, but real pilots know that this is a crucial maneuver to improve the passage speed of the test. Re-discovering this experimental knowledge thanks to our simulations is really satisfying and strengthens the credibility of our result.

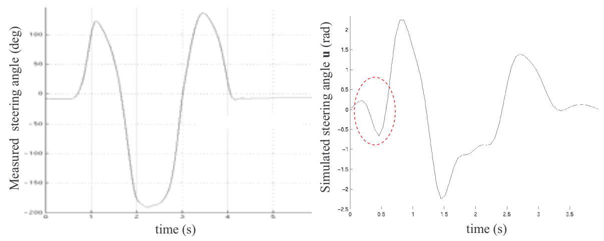

Fig 12 Comparison between a measured and a simulated steering angle

Despite these encouraging results we cannot give any formal guarantee that the maximum speed we reached is really the right one, or that the steering profile we discovered is the optimal one. Indeed, the algorithm can be trapped in a local minimum and give wrong results. Nevertheless the exploration algorithms of II provide good initial solutions and makes us confident in the quality of the results.

VI. Conclusions and perspectives

As a conclusion, this paper has presented a transfer of robotics research results to the domain of car design. The methodology we have proposed to compute trajectories that pass driving tests consists in computing a first coarse trajectory that pass the test at low speed and then to optimize this trajectory. The first step is performed using classical exploration methods, while the second step is performed using an extension to dynamic systems of an existing trajectory optimization method.

The main drawback of our approach is the rather long computing time induced by the numerical calculus of **A** and **B**, even though several computing tricks can improve it by an order of magnitude. Other limitations are the lack of formal guarantee of success and the border effects of the input perturbation functions.

Meanwhile, this powerful approach is truly generic and works for any dynamic system (even for black boxes). Therefore we plan to extend it to many other tests like a curve braking situation, i.e. a solicitation that can cause a strong instability of the vehicle. In this problem, there are no considerations of maximum passage speed but the objective is to find a representative set of solutions allowing the passage of the test. These data could help PSA Peugeot Citroën engineers to evaluate the quality of the behavior of a car in such critical situations, and contribute to find adequate solutions.

References

[1] J. Barraquand and J.-C. Latombe, Nonholonomic Multibody Mobile Robots: Controllability and Motion Planning in the Presence of Obstacles. Algorithmica, vol 10, pp 121--155, 1993.

[2] S.M. LaValle and J.J. Kuffner, Randomized kinodynamic planning. International Journal of Robotics Research, 20(5):378-400, May 2001.

[3] R. Kindel, D. Hsu, J.C. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. IEEE Int. Conf. Robot. & Autom., April 2000.

[4] D. Hsu, J-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. IEEE Int. Conf. on Robotics and Automation (ICRA), pages 2719--2726, 1997.

[5] P. Cheng and S. M. LaValle. Reducing metric sensitivity in randomized trajectory design. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pages 43--48, 2001.

[6] Quinlan, S. and Khatib, O. Elastic bands: Connecting path planning and control. IEEE Int. Conf. on Robotics and Automation (ICRA)," pages 802--807, Atlanta, 1993.

[7] John T. Bets, Survey of numerical methods for trajectory optimization, Journal of guidance, control and dynamics, vol 21, No.2, March-April 1998.

[8] F. Lamiraux, D. Bonnafous and O. Lefebvre, Reactive Path Deformation for Nonholonomic Mobile Robots. IEEE Transactions on Robotics, vol 20, No 6, pp 967-977, December 2004.

[9] Eric Larsen, Stefan Gottschalk, Ming C. Lin, Dinesh Manocha, Fast Proximity Queries with Swept Sphere Volumes, Technical report TR99-018, Department of Computer Science, University of N. Carolina, Chapel Hill, http://www.cs.unc.edu/~geom/SSV