Reactive Trajectory Deformation To Navigate Dynamic Environments

Vivien Delsart and Thierry Fraichard

Inria Rhône-Alpes, LIG-CNRS, Grenoble Universities (FR)

Abstract. Path deformation is a technique that was introduced to generate robot motion wherein a path, that has been computed beforehand, is continuously deformed on-line in response to unforeseen obstacles. In an effort to improve path deformation, this paper presents a trajectory deformation scheme. The main idea is that by incorporating the time dimension and hence information on the obstacles' future behaviour, quite a number of situations where path deformation would fail can be handled. The trajectory represented as a space-time curve is subject to deformation forces both external (to avoid collision with the obstacles) and internal (to maintain trajectory feasibility and connectivity). The trajectory deformation scheme has been tested successfully on a planar robot with double integrator dynamics moving in dynamic environments.

1 Introduction

Where to move next? is a key question for an autonomous robotic system. This fundamental issue has been largely addressed in the past forty years. Many motion determination strategies have been proposed (see [1] for a review). They can broadly be classified into *deliberative* versus *reactive* strategies: deliberative strategies aim at computing a complete motion all the way to the goal, whereas reactive strategies determine the motion to execute during the next few time-steps only. Deliberative strategies have to solve a motion planning problem. They require a model of the environment as complete as possible and their intrinsic complexity is such that it may preclude their application in dynamic environments. Reactive strategies on the other hand can operate on-line using local sensor information: they can be used in any kind of environment whether unknown, changing or dynamic, but convergence towards the goal is difficult to guarantee.

To bridge the gap between deliberative and reactive approaches, a complementary approach has been proposed based upon *motion deformation*. The principle is simple: a complete motion to the goal is computed first using a priori information. It is then passed on to the robotic system for execution. During the course of the execution, the still-to-be-executed part of the motion is continuously deformed in response to sensor information acquired on-line, thus accounting for the incompleteness and inaccuracies of the a priori world model. Deformation usually results from the application of constraints both external (imposed by the obstacles) and internal (to maintain motion feasibility and connectivity). Provided that the motion connectivity can be maintained, convergence towards the goal is achieved.

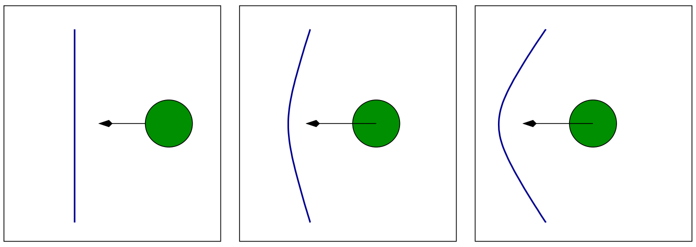


Fig. 1: Path deformation problem: in response to the approach of the moving disk, the path is increasingly deformed until it snaps (like an elastic band).

The different motion deformation techniques that have been proposed [2–6] all performs *path deformation*. In other words, what is deformed is a geometric curve, *ie* the sequence of positions that the robotic system is to take in order to reach its goal. The problem with path deformation techniques is that, by design, they cannot take into account the time dimension of a dynamic environment. For instance in a scenario such as the one depicted in Fig. 1, it would be more appropriate to leave the path as it is and adjust the velocity of the robotic system along the path so as to avoid collision with the moving obstacle (by slowing down or accelerating). To achieve this, it is necessary to depart from the path deformation paradigm and resort to *trajectory deformation* instead. A trajectory is essentially a geometric path parameterized by time. It tells us where the robotic system should be but also when and with what velocity. Unlike path deformation wherein spatial deformation only takes place, trajectory deformation features both *spatial and temporal* deformation meaning that the planned velocity of the robotic system can be altered thus permitting to handle gracefully situations such as the one depicted in Fig. 1.

The first trajectory deformation scheme has been proposed by one of the authors in [7]. It operates in two stages (collision avoidance and connectivity maintenance stages) and was limited to holonomic robotic systems. The contribution of this paper is a new trajectory deformation scheme, henceforth called `Teddy` (for Trajectory Deformer). It operates in one stage only and is explicitly designed to handle arbitrary nonholonomic and dynamic constraints.

The paper is organised as follows: `Teddy` is overviewed in §2. Its application to the case of a planar robot with double integrator dynamics (subject to velocity and acceleration bounds) is detailed in §3. Experimental results are then presented in §4.

2 Overview of the Approach

2.1 Notations and Definitions

Let $\mathcal{A}$ denote a robotic system operating in a workspace $\mathbf{W}$ ($\mathbb{R}^2$or $\mathbb{R}^3$). $q \in \mathbf{C}$ denote a configuration of $\mathcal{A}$. The dynamics of $\mathcal{A}$ is described by a differential equation of the form:

$$\dot{s} = f(s, u)$$

where $s \in \mathbf{S}$ is the state of $\mathcal{A}$, $\dot{s}$ its time derivative and $u \in \mathbf{U}$ a control. $\mathbf{C}$, $\mathbf{S}$ and $\mathbf{U}$ respectively denote the configuration space, the state space and the control space of $\mathcal{A}$. Let $\xi : [0, t_f[\longrightarrow \mathbf{U}$ denote a control input, *ie* a time-sequence of controls. Starting from an initial state s_0 (at time 0) and under the action of a control input ξ, the state of $\mathcal{A}$ at time t is denoted by $\xi(s_0, t)$. A couple (s_0, ξ) defines a trajectory for $\mathcal{A}$, *ie* a curve in $\mathbf{W} \times \mathbf{T}$.

For the sake of trajectory deformation, a trajectory will be discretized in a sequence of nodes. A node is a state-time, it is denoted by $n_i = (s_i, t_i)$. The discrete trajectory of $\mathcal{A}$ is $\Gamma_0 = \{n_0, n_1 \cdots n_N\}$ with n_0 (resp. n_N) the initial (resp. final) node of the trajectory.

2.2 Trajectory Deformation Principle

`Teddy` operates periodically: at time t_k, it takes as input the still-to-be-executed part of the trajectory $\Gamma_k = \{n_k, n_{k+1} \cdots n_N\}$ and an updated model of the workspace. This model includes the position of the obstacles of $\mathbf{W}$ at time t_k along with information about their future behaviour. `Teddy` then deforms Γ_k in response to the updated position and future behaviour of the obstacles and outputs a deformed trajectory $\Gamma_k{}' = \{n_k, n_{k+1}{}' \cdots n_N{}'\}$ with $n_i{}'$ the updated node corresponding to n_{k+1}.

Like a particle placed in a force field, a node is displaced in response to the application of a force which is the combination of two kind of forces: external and internal. External forces are repulsive forces exerted by the obstacles of the environment, their purpose is to deform the trajectory in order to keep it collision-free. They are detailed in §2.3. Internal forces on the other hand are aimed at maintaining the feasibility and the connectivity of the trajectory, *ie* to ensure that the deformed trajectory still satisfies the kinematic and dynamic constraints of $\mathcal{A}$. They are detailed in §2.4.

In certain cases, the constraints imposed by the environment are such that the deformation process fails to produce a trajectory which remains collision-free and connected (for instance when the topology of $\mathbf{S} \times \mathbf{T}$ changes). Such failures cannot generally be avoided since the topological information provided by the initial trajectory becomes invalid. Should the situation arise, a motion planner must be invoked to compute a new trajectory.

2.3 External Forces

External forces are repulsive forces exerted by the obstacles of the environment for collision avoidance purposes. They are derived from a potential function V_{ext}. To explicitly take into account the future behaviour of the moving obstacles, V_{ext} is defined in the space-time $\mathbf{W} \times \mathbf{T}$ (instead of $\mathbf{S} \times \mathbf{T}$ for efficiency reason). In a manner similar to [4], a set of points p_j are selected on the body of $\mathcal{A}$. Each node n_i of the trajectory Γ_k yield a set of control points $c_i^j = (p_j, t_i)$ in $\mathbf{W} \times \mathbf{T}$. For a control point c corresponding to the configuration q and the state s along the trajectory, V_{ext} is defined as:

$$V_{ext}(c) = \begin{cases} k_{ext}(d_0 - d_{wt}(c))^2 & \text{if } d_{wt}(c) < d_0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $d_{wt}(c)$ is the distance from c to the closest obstacle in $\mathbf{W} \times \mathbf{T}$. d_0 is the region of influence around the obstacles and k_{ext} is the repulsion gain. d_{wt} is a distance function in $\mathbf{W} \times \mathbf{T}$. It is derived from the Euclidean distance by scaling the space versus the time dimension. In $\mathbb{R}^2$ for instance, the distance d_{wt} between (x_0, y_0, t_0) and (x_1, y_1, t_1) is given by:

$${d_{wt}}^2 = {w_s}^2(x_1 - x_0)^2 + {w_s}^2(y_1 - y_0)^2 + {w_t}^2(t_1 - t_0)^2 \tag{2}$$

with w_s (resp. w_t) the spatial (resp. temporal) weight. The force resulting from this potential function acting on c is then defined as:

$$\mathbf{F}_{ext}^{wt}(c) = -\nabla V_{ext}(c) = k_{ext}(d_0 - d(c))\frac{\mathbf{d}}{||\mathbf{d}||} \tag{3}$$

where $\mathbf{d}$ is the vector between c and the closest obstacle point. Now, $\mathbf{F}_{ext}^{wt}$ has to be mapped into $\mathbf{S} \times \mathbf{T}$. $\mathbf{F}_{ext}^{wt}$ is first mapped into $\mathbf{C} \times \mathbf{T}$ as follows:

$$\mathbf{F}_{ext}^{ct} = J_c^T(q, t)\mathbf{F}_{ext}^{wt}(c) \tag{4}$$

where $J_c^T(q, t)$ represents the Jacobian at point c. The mapping into $\mathbf{S} \times \mathbf{T}$ that yields $\mathbf{F}_{ext}$ is carried out by leaving the additional state parameters unchanged.

2.4 Internal Forces

The external forces defined above push each node of the trajectory away from the obstacles if they are inside their influence region. Internal forces are introduced to ensure that the trajectory remains connected, *ie* that there exists a trajectory verifying the dynamics of $\mathcal{A}$ between two consecutive nodes of the trajectory. Trajectory connectivity is related to the concepts of forward and backward reachability. The set of states that are reachable from a given state s_0 are defined as:

$$\mathcal{R}(s_0) = \{s \in \mathbf{S} | \exists \xi, \exists t, \xi(s_0, t) = s\} \tag{5}$$

Likewise, the set of states from which it is possible to reach a given state s_0 are defined as:

$$\mathcal{R}^{-1}(s_0) = \{s \in \mathbf{S} | \exists \xi, \exists t, \xi(s, t) = s_0\} \tag{6}$$

Let n_-, n and n_+ denote three consecutive nodes of the trajectory $\mathit{\Gamma}_k$. $\mathit{\Gamma}_k$ is connected at n iff $n \in \mathcal{R}(n_-)$ and $n_+ \in \mathcal{R}(n)$. In other words, n must belong to $\mathcal{R}(n_-) \cap \mathcal{R}^{-1}(n_+)$. The purpose of the internal forces is to ensure that n remains within $\mathcal{R}(n_-) \cap \mathcal{R}^{-1}(n_+)$. To that end, a virtual spring is defined between n and H, the centroid of $\mathcal{R}(n_-) \cap \mathcal{R}^{-1}(n_+)$. It yields a potential function V_{int} defined in the space-time $\mathbf{S} \times \mathbf{T}$ as:

$$V_{int}(n) = k_{int} d_{st}(n)^2 \tag{7}$$

where $d_{st}(n)$ is the distance between n and H. It is defined in a manner similar to d_{wt}. k_{int} is the attraction gain.

$$\mathbf{F}_{int}(n) = -\nabla V_{int}(n) = k_{int} d(n) \frac{\mathbf{d}}{||\mathbf{d}||} \tag{8}$$

where $\mathbf{d}$ is the vector between n and H.

2.5 Total Force

Once both internal and external forces have been computed for a node n, the net force applied to it is:

$$\mathbf{F}(n) = \mathbf{F}_{ext}(n) + \mathbf{F}_{int}(n) \tag{9}$$

3 Case Study: Double Integrator System

To begin with, `Teddy` has been applied to the case of a 2D planar robot $\mathcal{A}$ with double integrator dynamics. A state of $\mathcal{A}$ is characterised by (p, v) that respectively denote the 2D position and velocity of $\mathcal{A}$ ($|v| \leq v_{\max}$). The dynamics of $\mathcal{A}$ is given by:

$$\begin{pmatrix} \dot{p} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} v \\ a \end{pmatrix} \tag{10}$$

with a the acceleration control applied to $\mathcal{A}$ ($|a| \leq a_{\max}$).

The key point in the adaptation of `Teddy` to a particular robotic systems lies in the computation of $\mathcal{R}(n_-) \cap \mathcal{R}^{-1}(n_+)$ and its centroid H. Fig. 2 depicts an example of a region $\mathcal{R}(n_-) \cap \mathcal{R}^{-1}(n_+)$ and its centroid H obtained by numerical computation in the 1D case. Depending on the robotic system considered, this computation can be too complex to allow `Teddy` to operate reactively. To address this issue, it was decided to compute conservative approximations of $\mathcal{R}(n_-)$ and $\mathcal{R}^{-1}(n_+)$ using an efficient approximation scheme inspired by [8]. The basic principle is to compute the convex hull of both (n_-) and $\mathcal{R}(n_-, t_+)$ with $\mathcal{R}(n_-, t_+)$ the t_+-slice of $\mathcal{R}(n_-)$. To ensure the conservativeness of the approximation, both (n_-) and $\mathcal{R}(n_-, t_+)$ must be grown appropriately (the reader interested in this part is referred to [9] or [10] for the English version).

To speed things up, the basic entity which is used to perform all geometric computations is the zonotope which is a particular class of convex polytopes for

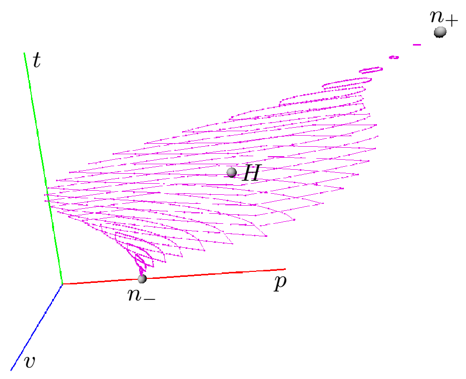


Fig. 2: $\mathcal{R}(n_-) \cap \mathcal{R}^{-1}(n_+)$ and its centroid H in the 1D case.

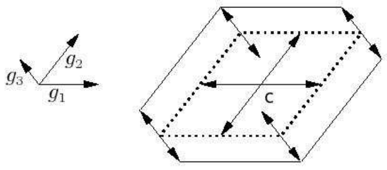


Fig. 3: The 2D zonotope defined by the center c and the vectors g_1, g_2, g_3.

which Minkowski operations can be performed very effciently [11]. A zonotope $\mathcal{Z}$ is defined as:

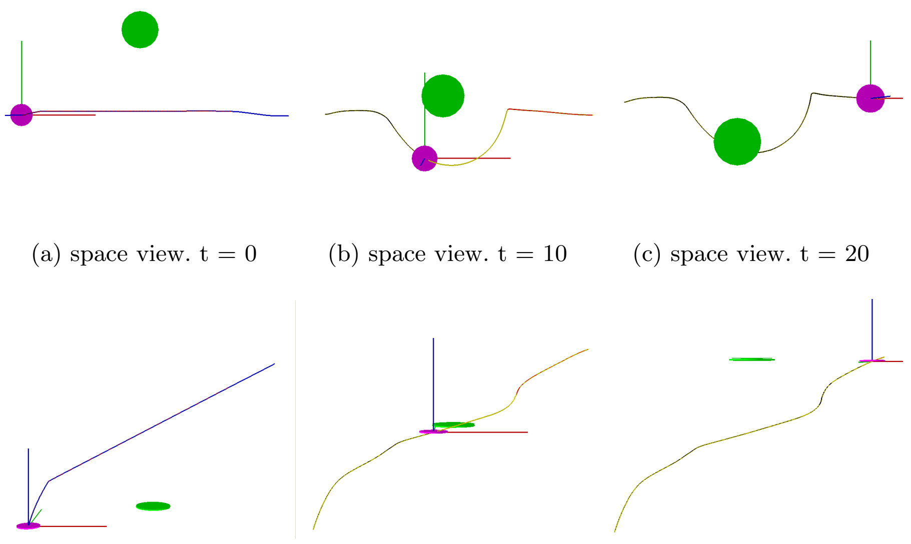
$$\mathcal{Z} = \left\{ x \in \mathbb{R}^n : x = c + \sum_{i=1}^{i=p} x_i g_i, -1 \leq x_i \leq 1 \right\} \tag{11}$$

with c the center of $\mathcal{Z}$, and $\{g_1 \cdots g_n\}$ a set of direction vectors (Fig. 3).

4 Experimental Results

`Teddy` has been implemented in C++ and tested on an Intel Pentium 4 desktop PC (3GHz, 1GB RAM, Linux OS). `Teddy` has been evaluated in different scenarios featuring up to 10 circular obstacles moving randomly. At each time step, `Teddy` is provided with a new model of the environment and its future evolution. To better illustrate, the interest of `Teddy`, we have focused in this section on a simple "cutting" scenario similar to the one depicted in Fig. 1. This scenario has been selected because it is problematic for classical path deformation schemes.

`Teddy` relies upon a number of parameters to operate properly: the repulsion gain k_{ext}, the attraction gain k_{int} and the distance functions d_{wt} and d_{st}. The two examples presented below have been selected to illustrate the importance of



(a) space view. t = 0 (b) space view. t = 10 (c) space view. t = 20

(d) time view. t = 0 (e) time view. t = 10 (f) time view t = 20

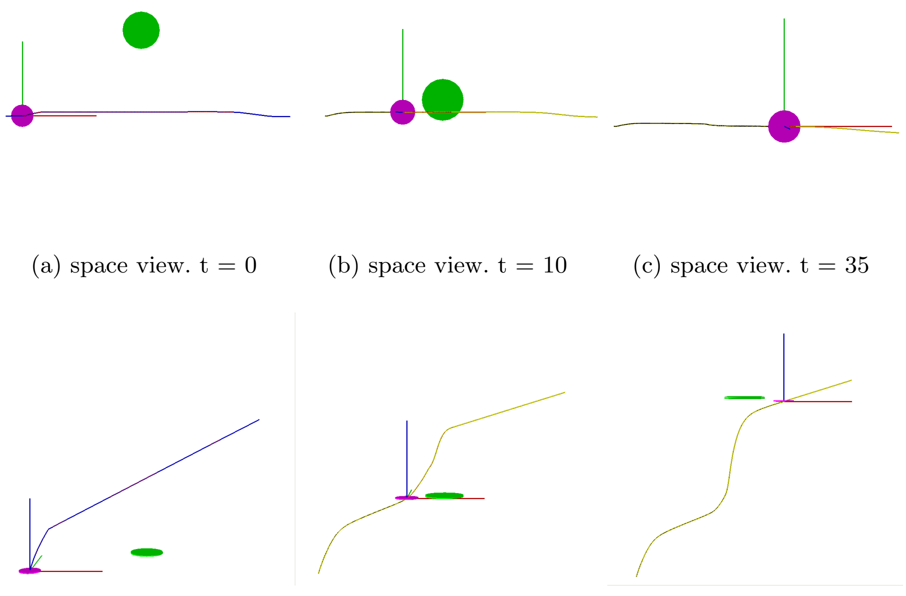
Fig. 4: Example 1 (spatial deformation): $\mathcal{A}$ is moving from the left to the right, the obstacle is moving downwards. The top snapshots depict the path at different time instant ($x \times y$ view). The bottom snapshots depict the velocity profile at the same instants ($x \times t$ view).

the distance function d_{wt} on the performance of `Teddy`. Recall that d_{wt} is used to determine the distance between a trajectory node and the closest obstacle in $\mathbf{W} \times \mathbf{T}$ (*cf* §2.3). In both examples, the initial trajectory had a duration of 20s and the discrete trajectory contained 320 nodes. `Teddy` would run at approximately 28Hz.

In the same situation, two very different deformation patterns can be obtained by properly selecting the weights w_s and w_t in (2). The first example is obtained by giving more weight to w_s thereby allowing more important spatial deformations to take place (Fig. 4). In this case, $\mathcal{A}$ has time to pass before the obstacle crosses its path. The path component of the trajectory is deformed downwards for safety reasons whereas the velocity component is only slightly modified.

The second example on the other hand is obtained by giving more weight to w_t thereby allowing more important temporal deformations to take place (Fig. 5). In this case, $\mathcal{A}$ let the obstacle cross its path before proceeding. The path component of the trajectory is only slightly modified whereas the velocity component is largely deformed so as to allow $\mathcal{A}$ to slow down and stop in order to give way to the obstacle.

These two examples have shown the influence of the choice of the parameters in the final performance of `Teddy`. They have also illustrated the advantage of



(a) space view. t = 0 (b) space view. t = 10 (c) space view. t = 35

(d) time view. t = 0 (e) time view. t = 10 (f) time view t = 35

Fig. 5: Example 2 (temporal deformation): $\mathcal{A}$ is moving from the left to the right, the obstacle is moving downwards. The top snapshots depict the path at different time instant ($x \times y$ view). The bottom snapshots depict the velocity profile at the same instants ($x \times t$ view).

number of	number of nodes				
obstacles	50	100	180	250	320
1	6	11	20	27	35
3	44	48	68	70	73
10	49	88	135	199	229

Table 1: Running time (in ms) of one deformation cycle as a function of the number of nodes and obstacles.

trajectory deformation versus path deformation. From a complexity point of view, it grows linearly with the number of nodes and the number of obstacles. Table 1 gives the running time of one deformation cycle for different numbers of nodes and obstacles.

5 Conclusion and Future Works

The paper has presented `Teddy`, a trajectory deformation scheme. Given a nominal trajectory reaching a given goal, `Teddy` deforms it reactively in response to updated information about the environment's obstacles. Teddy can handle robotic systems with arbitrary dynamics. It has been applied to the case of a 2D double integrator system and tested in various situations. Because, Teddy explicitly takes into account information on the future behaviour of the obstacles, it is able to handle situations that are problematic for classical path deformation schemes. In the future, it is planned to consider other robotic systems, *eg* car-like vehicles, and further optimize Teddy. Considering for instance that the knowledge about the future behaviour is less reliable in the distant future, it could be interesting to monotonically decrease the influence of the obstacles with respect to time. Last but not least, Teddy remains to be integrated within a global navigation architecture and tested on an actual robotic system. It is planned to do so on the architecture and the vehicle presented in [12]

References

1. Lavalle, S.M.: Planning Algorithms. Cambridge University Press (2006)
2. Quinlan, S., Khatib, O.: Elastic bands: Connecting path planning and control. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, Atlanta, GA (US) (May 1993)
3. Khatib, M., Jaouni, H., Chatila, R., Laumond, J.P.: Dynamic path modification for car-like nonholonomic mobile robots. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, Albuquerque, NM (US) (April 1997)
4. Brock, O., Khatib, O.: Elastic strips: a framework for motion generation in human environments. Int. Journal of Robotics Research **21**(12) (December 2002)
5. Lamiraux, F., Bonnafous, D., Lefebvre, O.: Reactive path deformation for nonholonomic mobile robots. IEEE Trans. on Robotics and Automation **20**(6) (December 2004)
6. Yang, Y., Brock, B.: Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation. In: Proc. of the Int. Conf. Robotics: Science and Systems, Philadelphia PA (US) (August 2006)
7. Kurniawati, H., Fraichard, T.: From path to trajectory deformation. In: Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, San Diego, CA (US) (October 2007)
8. Dang, T.: Vérification et synthèse des systèmes hybrides. Thèse de doctorat d'état, Institut National Polytechnique de Grenoble, Grenoble (FR) (2000)
9. Delsart, V.: Autonomie du mouvement en environnement dynamique. Master report, Inst. Nat. Polytechnique, Grenoble (FR) (June 2007)
10. Delsart, V., Fraichard, T.: Reactive trajectory deformation. Research report, INRIA (In Press)
11. Girard, A.: Reachability of uncertain linear systems using zonotopes. In Morari, M., Thiele, L., eds.: Hybrid Systems : Computation and Control. Volume 3414 of Lecture Notes in Computer Science. Springer (2005)
12. Chen, G., Fraichard, T., Martinez-Gomez, L.: A real-time autonomous navigation architecture. In: Proc. of the IFAC Symp. on Intelligent Autonomous Vehicles, Toulouse (FR) (September 2007)