Proceedings of the 1992 IEEE
International Conference on Robotics and Automation
Nice, France - May 1992

Kinodynamic planning in a structured and time-varying 2D workspace*

Th. Fraichard and C. Laugier†
LIFIA-IRIMAG
46, av. Félix Viallet, 38031 Grenoble Cedex, France

Abstract

This paper deals with a trajectory planning problem that we call the 'highway problem'. It consists in planning a time-optimal trajectory for a mobile which is travelling in a structured workspace amidst moving obstacles and is subject to constraints on its velocity and acceleration. By structured workspace, we mean that there exists lanes characterized by one-dimensional curves along which the mobile is able to move. The mobile has to follow a lane but it may also shift from its lane to an adjacent one.

This paper presents an efficient method which determines an approximate time-optimal solution to the highway problem. The approach consists in discretizing time and selecting the accelerations applied to the mobile among a discrete set. These hypotheses make it possible to define a grid in the mobile's time-state space, i.e. the mobile's state (or phase) space augmented of the time dimension. This grid is then searched in order to find a solution.

1 Introduction

1.1 Overview of the problem

Planning motions for robots is a fundamental problem which encompasses a wide range of approaches and assumptions. However it seems important to make a distinction between *path planning* and *trajectory planning*: path planning is characterized by the search of a continuous sequence of collision-free configurations between a start and a goal configuration, whereas trajectory planning is concerned with the time history of this sequence of configurations.

This paper deals with a particular trajectory planning problem that we call the 'highway problem'. It consists in planning a time-optimal trajectory for a mobile which is travelling in a structured workspace amidst moving obstacles and is subject to constraints on its velocity and acceleration. By structured workspace, we mean that there exists lanes characterized by one-dimensional curves along which the mobile is able to move. The mobile has to follow a lane but it may also shift from its lane to an adjacent one. Such lanes may be a priori defined by the intrinsic structure of the workspace, as in [8], but they may also be automatically extracted from a description of the workspace, as in [2].

Planning the motion of a car on the highway among other cars is a vivid example of this kind of problem, hence the name 'highway problem'. This framework led us to take into account another constraint: in a workspace such as the roadway, there are several potential moving obstacles (cars, pedestrians...) and it is impossible to have a full a priori knowledge of their motions. Therefore we will assume that the knowledge that we have of the motions of the moving obstacles is restricted to a certain time interval—the *time-horizon*. The main consequence of this assumption is to define an upper bound on the time available to plan the motion of the mobile considered.

1.2 Contribution of the paper

This paper presents an efficient method which solves the highway problem. The approach which we have chosen was initially motivated by the work described in [4]: it consists in discretizing time and selecting the accelerations applied to the mobile among a discrete set. These hypotheses make it possible to define a grid in the mobile 's state (or phase) space. This grid is then searched in order to find a solution. In our case however, moving obstacles have to be dealt with, therefore we will define the grid in the *time-state space*[1] of the mobile rather than in its state space. As we will see further down. the choice of the time-step determines the size of the grid which, in turn, determines the running time of the algorithm as well as how close the solution is to the exact time-optimal solution.

*This work was supported by the French Ministry of Research and Technology and the European Prometheus Eurêka project.

†Research director at INRIA. tf / laugier@lifia.imag.fr

[1] The time dimension is explicitly added to the mobile's state space.

0-8186-2720-4/92 $3.00 ©1992 IEEE

The paper is organized as follows: §2 briefly reviews the complexity issues and the works related to trajectory planning. Then §3 formally states the highway problem. §4 describes the algorithm developped in order to solve this problem while §5 presents an implementation of this algorithm along with experimental results.

2 Complexity issues and related works

2.1 Complexity of the problem

The 'piano mover' problem —i.e. path planning — is well-known for its complexity. Various instances of this problem have been shown to be Pspace-hard, Pspace-complete or NP-hard (see [12]). Since trajectory planning takes into account an extra dimension — time— and extra dynamic constraints, one can expect it to be as computationally expensive as path planning, and it is indeed. The various results presented in [3,13] or [14] indicate that the highway problem is intricate. Therefore, in order to have a chance to meet the time-horizon constraint, a few simplifying assumptions are required. These assumptions rely upon the specific structure of the workspace considered. They are presented in §3 along with the formal statement of the problem.

2.2 Related works

There is a large body of works related to trajectory planning. Some of them focus on trajectory planning amidst moving obstacles while others deal more particularly with dynamic constraints.

Moving obstacles: a general approach in order to deal with moving obstacles is the so-called 'configuration-time space' approach which consists in adding the time dimension to the robot's configuration space. (cf [7, 9, 14, 16]). An alternate approach is the 'path-velocity decomposition'. The basic idea is to decompose the trajectory planning into two subproblems: (a) planning a path which avoids collision with the static obstacles of the workspace and (b) planning the velocity along this path in order to avoid collision with the moving obstacles. This approach was first introduced in [11]. It is efficient but, unfortunately, it is inherently incomplete.

Dynamic constraints: dealing with dynamic constraints has proved to be an intricate problem. There are some results for exact time-optimal trajectory planning for Cartesian robots subject to bounds on their velocity and acceleration [13, 5]. Besides optimal control theory provides some exact results in the case of robots with full dynamics moving along a given path [1, 18]. However the difficulty of the general problem and the need for practical algorithms led some authors to develop approximate methods. Their basic principle is to define a grid which is searched in order to find an optimal solution. Accordingly trajectory planning is reduced to graph search (cf [4, 6, 10, 17, 15]).

3 Statement of the problem

The lanes which represent the structure of the workspace are modelled by a set of l straight planar curves L_i, $i = 0, 2 \ldots l-2, l$ (the reason for this peculiar indexing will appear later). The L_i have an equal length p_{max} and are arranged in the way depicted in figure 1. A lane will be referred to by its index.

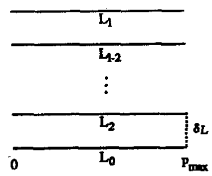

Figure 1: the lanes L_i, $i = 0, 2 \ldots l-2, l$

A set of particles $\mathcal{B}_j$, $j = 1 \ldots n$, represents the obstacles to be avoided. Wlog, it is assumed that they are moving along the lanes. The position of such a particle at time t is defined by the tuple $(L_{\mathcal{B}_j}(t), p_{\mathcal{B}_j}(t))$ where $L_{\mathcal{B}_j}(t)$ is the lane of $\mathcal{B}_j$ and where $p_{\mathcal{B}_j}(t) \in [0,\ p_{max}]$ is its curvilinear abcissa along this lane. The functions $L_{\mathcal{B}_j}(t)$ and $p_{\mathcal{B}_j}(t)$ are defined (but not necessarily everywhere) over the closed time interval $[0,\ t_{max}]$ where t_{max} represents the time-horizon, i.e. the time interval over which the motions of the $\mathcal{B}_j$ are known.

The mobile whose motions are to be planned is also represented by a particle $\mathcal{A}$. The normal behaviour of $\mathcal{A}$ is to follow a given lane; however $\mathcal{A}$ has also the possibility to make a lane-changing, i.e. to shift from its current lane to an adjacent one. The particular nature of these motions led us to decouple the motion along the lane —*longitudinal motion*— from the motions between the lanes —*lateral motions*.

Let us consider the lateral motion, i.e. the lane-changing depicted in figure 2-a. At time t, $\mathcal{A}$ shifts smoothly from its current lane L_i to an adjacent lane L_{i+2}. Let Δt be the time interval necessary to perform the lane-changing. The shape of the lane-changing trajectory and the value of Δt depend on the characteristics of $\mathcal{A}$. Assuming that the value of δL is of the order of the width of the vehicles represented by

$\mathcal{A}$ and the $\mathcal{B}_j$[2], it is reasonable that during the lane-changing, $\mathcal{A}$ should avoid collision with the obstacles of both lanes L_i and L_{i+2}. This property makes it possible to model the lateral motion as a three-step process: (a) at time t, $\mathcal{A}$ instantaneously 'jumps' from L_i to a fictitious intermediate lane L_{i+1}, (b) $\mathcal{A}$ moves along L_{i+1} during $[t, t+\Delta t]$ (the obstacles of both L_i and L_{i+2} are assumed to be on L_{i+1}) and (c) at time $t+\Delta t$, $\mathcal{A}$ instantaneously 'jumps' from L_{i+1} to L_{i+2} (see figure 2-b). Accordingly this modelling reduces a lateral motion to a longitudinal motion along a fictitious lane.

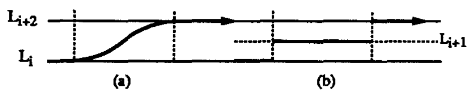

Figure 2: lane-changing

The longitudinal motion of $\mathcal{A}$ along a real or fictitious lane is a one-dimensional motion which is obtained by applying an acceleration $\ddot{p}(t)$ to $\mathcal{A}$. The velocity $\dot{p}(t)$ and the position $p(t)$ of $\mathcal{A}$ along this lane are respectively defined as the first and second integral of $\ddot{p}(t)$ subject to an initial position and an initial velocity. Besides $\ddot{p}(t)$ and $\dot{p}(t)$ are bounded:

$$-\ddot{p}_{max} \leq \ddot{p}(t) \leq \ddot{p}_{max} \quad (1)$$
$$0 \leq \dot{p}(t) \leq \dot{p}_{max} \quad (2)$$

In this framework, a *state* of $\mathcal{A}$ is defined as being the tuple $(L, p, \dot{p})$ where $L \in \{0 \dots l\}$ is the index of the current lane of $\mathcal{A}$, $p \in [0, p_{max}]$ its curvilinear abcissa along this lane and $\dot{p} \in [0, \dot{p}_{max}]$ its instantaneous velocity. Accordingly a *trajectory* for $\mathcal{A}$ is defined by a mapping Γ taking a time $t \in [0,\ t_f]$ to a state $\Gamma(t) = (L(t), p(t), \dot{p}(t))$. The *time* for the trajectory Γ is simply t_f. The two components $L(t)$ and $(p(t), \dot{p}(t))$ of this trajectory are defined by the two following maps:

1. L: $[0,\ t_f] \longrightarrow \{0 \dots l\}$ which indicates the current lane of $\mathcal{A}$. If L_t is even then $\mathcal{A}$ is on the real lane L_t, otherwise it is performing a lane-changing between the lanes $L_t - 1$ and $L_t + 1$.
2. $\ddot{p}$: $[0,\ t_f] \longrightarrow [-\ddot{p}_{max},\ \ddot{p}_{max}]$ which is the instantaneous acceleration applied to $\mathcal{A}$.

As to collision avoidance, a trajectory Γ is theoretically collision-free iff $\forall t \in [0,\ t_f], p_{\mathcal{B}_j}(t) \neq p(t)$ for every $\mathcal{B}_j$ such that $L_{\mathcal{B}_j}(t) = L(t)$. However, in order to have a trajectory which be of a certain practical value, $\mathcal{A}$ should avoid the obstacles $\mathcal{B}_j$ by a velocity-dependent safety margin $\delta(t)$. The purpose of this safety margin $\delta(t)$ is twofold —it generates a trajectory which (a) takes into account the fact that $\mathcal{A}$ and the $\mathcal{B}_j$ represent real sized vehicles and (b) is 'robust' in the sense that it does not skim over the obstacles (by doing so tracking errors will be allowed at execution time). Formally, a trajectory Γ is said to be *safe* iff:

$$\begin{aligned} &\forall t \in [0,\ t_f], \forall \mathcal{B}_j, j = 1 \dots n : \\ &\qquad L_{\mathcal{B}_j}(t) \neq L(t) \quad \text{or} \\ &\qquad L_{\mathcal{B}_j}(t) = L(t) \quad \text{and } |p(t) - p_{\mathcal{B}_j}(t)| > \delta(t) \end{aligned}$$

where $\delta(t) = c_0 + c_1 \mid \dot{p}(t) \mid$ with c_0 and c_1 two positive scalars.

Given an initial state $\mathbf{s} = (L_s, p_s, \dot{p}_s)$ and a final state $\mathbf{g} = (L_g, p_g, \dot{p}_g)$, a trajectory Γ constitutes a *solution* to the highway problem considered iff:

1. $L(0) = L_s$, $p(0) = p_s$, $\dot{p}(0) = \dot{p}_s$.
2. $\exists t_f \in [0,\ t_{max}]$ such that $L(t_f) = L_g$, $p(t_f) = p_g$, $\dot{p}(t_f) = \dot{p}_g$.
3. $\forall t \in [0,\ t_f], -\ddot{p}_{max} \leq \ddot{p}(t) \leq \ddot{p}_{max}$ and $0 \leq \dot{p}(t) \leq \dot{p}_{max}$.
4. $\forall t \in [0,\ t_f], L(t) \in \{\lambda,\ \lambda = 0 \dots 2l - 2\}$.
5. Γ is safe.

The problem to be solved is to find a *time-optimal solution*, i.e. a solution Γ defined for $t \in [0,\ t_f]$ such that t_f should be minimal.

4 The approach

4.1 The basic idea

In §3, we have defined the type of trajectory which is a solution to the highway problem. The intrinsic complexity of the problem (see §2.1) along with the time-horizon constraint led us to choose an approximate method to solve this problem. The approach consists in discretizing time —a time-step τ being chosen— and selecting the accelerations applied to the mobile from the discrete set $\{-\ddot{p}_{max},\ 0,\ \ddot{p}_{max}\}$. Accordingly, the solution trajectories that we consider meet the following constraints:

- $\ddot{p}(t) \in \{-\ddot{p}_{max},\ 0,\ \ddot{p}_{max}\}$ with $\ddot{p}$ piecewise constant.
- $L(t)$ and $\ddot{p}(t)$ only changes their values at times $t = k\tau$ for some integer $k \geq 0$.

[2] As is the case on a highway.

Such a trajectory will be called a *bang-trajectory*. Under these restrictions, the problem to be solved is to find out the time-optimal bang-trajectory. Obviously the complexity of this problem depends on the number of bang-trajectories which, in turn, is directly related to the size of τ —the smaller τ, the higher the number of bang-trajectories. On the other hand, we intuitively[3] feel that the closeness of the approximation is also related to the size of τ —the smaller τ, the better the approximation. Thus it is possible to trade off the computation speed against the quality of the solution.

In the next section, we show how to reduce the problem of finding out the time-optimal bang-trajectory to that of finding out a shortest path in a directed graph.

4.2 The time-state graph

A state of $\mathcal{A}$ has been defined earlier as being the tuple $(L, p, \dot{p})$. A *time-state* of $\mathcal{A}$ is defined by explicitely adding the time dimension to a state of $\mathcal{A}$. Let us denote TS the set of all such time-states. A point in TS is a tuple $s = (L, p, \dot{p}, t)$ or equivalently $s(t) = (L(t), p(t), \dot{p}(t))$.

Let $s(k\tau) = (L(k\tau), p(k\tau), \dot{p}(k\tau))$ be a time-state of $\mathcal{A}$ and let $s((k+1)\tau)$ be one of the time-states that $\mathcal{A}$ can reach by a bang-trajectory of duration τ. $s((k+1)\tau)$ is obtained by applying an acceleration $\ddot{p} \in \{-\ddot{p}_{max}, 0, \ddot{p}_{max}\}$ to $\mathcal{A}$ for the duration τ. Besides $\mathcal{A}$ has the possibility either to stay on its current lane or to switch to an adjacent lane. Accordingly we have:

$$\begin{aligned} L((k+1)\tau) &= L(k\tau) + \sigma \text{ with } \sigma \in \{-1, 0, 1\} \\ p((k+1)\tau) &= p(k\tau) + \dot{p}(k\tau)\tau + \frac{1}{2}\ddot{p}\tau^2 \\ \dot{p}((k+1)\tau) &= \dot{p}(k\tau) + \ddot{p}\tau \end{aligned}$$

As explained in §3, it is assumed that, during the time interval $]k\tau, (k+1)\tau]$, $\mathcal{A}$ is on the real or fictitious lane $L((k+1)\tau)$. By analogy with [4], the trajectory between $s(k\tau)$ and $s((k+1)\tau)$ is called a *$(\sigma, \ddot{p}, \tau)$-bang*. $s((k+1)\tau)$ is said to be *reachable* from $s(k\tau)$. Obviously a bang-trajectory is made up of a sequence of $(\sigma, \ddot{p}, \tau)$-bangs.

Let $s(m\tau)$, $m \geq k$, be a time-state reachable from $s(k\tau)$. Assuming that $\dot{p}(k\tau)$ is a multiple of $\ddot{p}_{max}\tau$, we can easily show that the following relations hold for some integers α_1, α_2 and α_3:

$$\begin{aligned} L(m\tau) &= L(k\tau) + \alpha_1\sigma \\ p(m\tau) &= p(k\tau) + \alpha_2\frac{1}{2}\ddot{p}_{max}\tau^2 \\ \dot{p}(m\tau) &= \dot{p}(k\tau) + \alpha_3\ddot{p}_{max}\tau \end{aligned}$$

Thus all time-states reachable from one given time-state by a bang-trajectory lie on a regular grid embedded in TS. This grid has spacings of τ in time, of $\frac{1}{2}\ddot{p}_{max}\tau^2$ in position, of $\ddot{p}_{max}\tau$ in velocity and of 1 in the lane dimension.

Consequently it becomes possible to define a directed graph $\mathcal{G}$ embedded in TS. The nodes of $\mathcal{G}$ are the grid-points while the edges of $\mathcal{G}$ are $(\sigma, \ddot{p}, \tau)$-bangs between pairs of these nodes. Such $(\sigma, \ddot{p}, \tau)$-bangs have to respect the velocity constraint and be safe (this point is detailed in §4.3). $\mathcal{G}$ is called the *time-state graph*, it is illustrated in figure 3 which depicts the 'time×position×velocity' space of two adjacent lanes L_i and L_{i+1}. For the sake of clarity, we have only represented one node and its neighbours on both lanes L_i and L_{i+1} (a node has at the most three neighbours per lane). Let A be this node. The time-states reachable from A by a $(\sigma, \ddot{p}, \tau)$-bang lie on the grid —they are nodes of $\mathcal{G}$. An edge between A and one of its neighbours represents the corresponding $(\sigma, \ddot{p}, \tau)$-bang. A sequence of edges between two nodes defines a bang-trajectory. The time of such a bang-trajectory is trivially equal to τ times the number of edges in the trajectory. Therefore the shortest path between two nodes *is* the time-optimal bang-trajectory between these nodes.

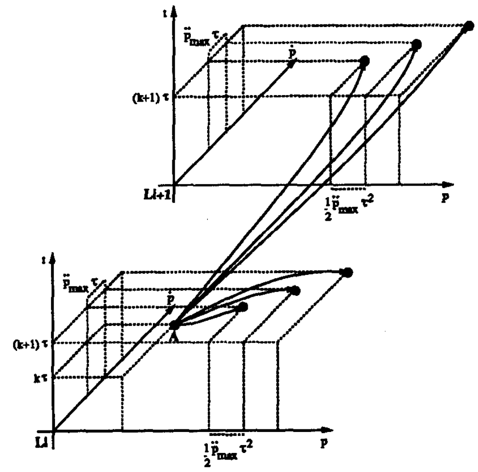

Figure 3: the graph $\mathcal{G}$

Let $s = (L_s, p_s, \dot{p}_s)$ and $g = (L_g, p_g, \dot{p}_g)$ be respectively the initial and the goal state for $\mathcal{A}$. Wlog, it is assumed that the time-state $s^* = (L_s, p_s, \dot{p}_s, 0)$ and

[3] This intuition is confirmed in [4] where it is shown that, for a correct choice of τ, any safe trajectory can be approximated to a tolerance ϵ by a safe bang-trajectory.

the set of time-states $\mathbf{G}^* = \{(L_g, p_g, \dot{p}_g, k\tau)$ with $k = 0 \ldots \lfloor \frac{t_{max}}{\tau} \rfloor\}$ are grid-points. Accordingly searching for a time-optimal bang-trajectory between $\mathbf{s}$ and $\mathbf{g}$ is equivalent to searching a shortest path in $\mathcal{G}$ between the node $\mathbf{s}^*$ and a node in $\mathbf{G}^*$.

Note that because we only consider a compact region of TS, the number of grid-points is finite. Thus $\mathcal{G}$ is finite and the search for the time-optimal bang-trajectory can be done in a finite amount of time. The next section describes how the search is carried out.

4.3 Searching the time-state graph

The algorithm: basically we use an A^* algorithm to search $\mathcal{G}$ Beside its efficiency, such an algorithm is interesting because it generates only the parts of $\mathcal{G}$ which are relevant to the search. Starting with $\mathbf{s}^*$ as the current node, we 'expand' this current node, i.e. we determine all its neighbours, then we select the neighbour which is the 'best' according to a given criterion and it becomes the current node. This process is repeated until the goal is reached or until the whole graph has been explored. The time-optimal path is returned using backpointers. In the next two sections, we detail two key-points of the algorithm. Namely the cost function assigned to each node and the node expansion.

The cost function: A^* assigns a cost $f(s)$ to every node s in $\mathcal{G}$. Since we are looking for a time-optimal path, we have chosen $f(s)$ as being the estimate of the time-optimal path in $\mathcal{G}$ connecting $\mathbf{s}^*$ to $\mathbf{G}^*$ and passing through s. $f(s)$ is classically defined as the sum of two components $g(s)$ and $h(s)$:

- $g(s)$ is the time of the path between $\mathbf{s}^*$ and s, i.e. the time component of s.
- $h(s)$ is the estimate of the time-optimal path between s and an element of $\mathbf{G}^*$, i.e. the amount of time it would take $\mathcal{A}$ to reach $\mathbf{g}$ from its current state with a 'bang-coast-bang' acceleration profile[4] in an obstacle-free workspace. When such an acceleration profile does not exist, $h(s)$ is set to $+\infty$.

The heuristic function $h(s)$ is trivially admissible, thus A^* is guaranteed to generate the time-optimal path whenever it exists.

The node expansion: the neighbours of a given node s in $\mathcal{G}$ are the nodes which can be reached from s by a $(\sigma, \ddot{p}, \tau)$-bang. As mentioned earlier, $\sigma \in \{-1, 0, 1\}$ and $\ddot{p} \in \{-\ddot{p}_{max}, 0, \ddot{p}_{max}\}$. Thus s has up to nine neighbours. However this number is reduced to three when $\mathcal{A}$ is performing a lane-changing (indeed $\mathcal{A}$ must stay on the fictitious intermediate lane for the duration Δt of the lane-changing). Let us consider the $(\sigma, \ddot{p}, \tau)$-bang between $s(k\tau)$ and $s((k+1)\tau)$ then, $\forall t \in [k\tau, (k+1)\tau]$, we have:

$$p(t) = p(k\tau) + \dot{p}(k\tau)(t - k\tau) + \frac{1}{2}\ddot{p}(t - k\tau)^2 \quad (3)$$

$$\dot{p}(t) = \dot{p}(k\tau) + \ddot{p}(t - k\tau) \quad (4)$$

This $(\sigma, \ddot{p}, \tau)$-bang is valid if it is safe and if it does not not violate the velocity bounds (2) introduced in §3. The velocity constraint can be tested easily using equation (4). As to collision avoidance, we must make sure that $\forall t \in [k\tau, (k+1)\tau]$, $p(t)$ is no closer than $\delta(t) = c_0 + c_1 \mid \dot{p}(t) \mid$ to any obstacle of the lane $L((k+1)\tau)$. $p(t)$ is computed using equation (3). A practical way to check out the safety of a given position $p(t)$ is to 'grow' of $\delta(t)$ the obstacles of the lane considered before testing whether $p(t)$ intersects the grown obstacles (see figure 4).

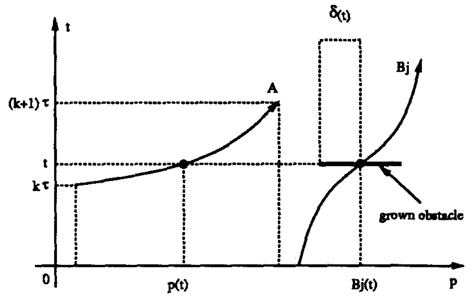

Figure 4: safety checking

5 Implementation and experiments

The algorithm presented above has been implemented in C on a Sun SPARC I. In the current implementation, the safety margin δ is constant and lane-changings are performed with a null acceleration.. Accordingly, a grid point has at the most five neighbours. We have tested the algorithm with up to four lanes. In these experiments, the obstacles are generated at random without caring whether they collide with each other. Besides they are assumed to keep a constant velocity over the time-horizon. An example of trajectory planning involving two lanes is depicted in figure 5. Each lane is associated with two windows: a trace window showing the part of $\mathcal{G}$ which has been explored

[4] i.e. maximum acceleration, null acceleration and maximum deceleration.

and a result window displaying the final trajectory. Any of these windows represents the 'time×position' space of the lane (the position axis is horizontal while the time axis is vertical; the frame origin is at the upper-left corner). The thick black segments represent the trails left by the moving obstacles and the little dots are points of the underlying grid. Note that the fictitious lane used to perform lane-changings is not represented here. $\mathcal{A}$ starts from the first lane (lane #0), at position 0 (upper-left corner) and with a null velocity. It must reach the first lane at position p_{max} (right border) with a null velocity. $\mathcal{A}$ can overtake by using the second lane (lane #1). In order to simulate the behaviour of a car on the roadway, we have chosen the following values for the various variables of the problem: $p_{max} = 500\text{m}$, $\delta L = 4\text{m}$, $\dot{p}_{max} = 72\text{km/h}$, $\ddot{p}_{max} = 1\text{m/s}^2$ et $t_{max} = 20\text{s}$. As mentionned earlier, it is the choice of τ which determines the average running time of the algorithm. For a value of τ set to 1s, we have obtained a running time ranging from less than a second to a few seconds.

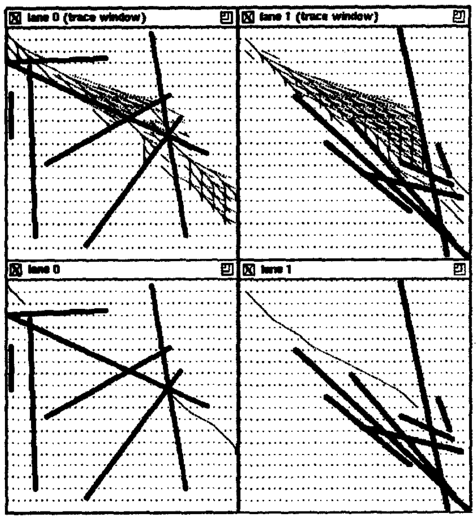

Figure 5:

References

[1] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. On the optimal control of robotic manipulators with actuator constraints. In *Proc. of the American Control Conf.*, pages 782–787, San-Francisco, CA (USA), June 1983.

[2] P. Caloud. Distributed motion planning and motion coordination for multiple robots. Working paper, Robotics Lab, Computer Science Dept, Stanford Univ, CA (USA), 1990.

[3] J. Canny. *The complexity of robot motion planning.* PhD thesis, Dept. of Electrical Engineering and Computer Science, MIT, Boston, MA (USA), 1987.

[4] J. Canny, B. Donald, J. Reif, and P. Xavier. On the complexity of kynodynamic planning. In *Proc. of the Symp. on the Foundations of Computer Science*, pages 306–316, White Plains, NY (USA), Nov. 1988. IEEE.

[5] J. Canny, A. Rege, and J. Reif. An exact algorithm for kinodynamic planning in the plane. In *Proc. of the Symp. on Computational Geometry*, pages 271–280, Berkeley, CA (USA), 1990. ACM.

[6] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open-chain manipulators. In *Proc. of the Symp. on Computational Geometry*, pages 290–300, Berkeley, CA (USA), 1990. ACM.

[7] M. Erdmann and T. Lozano-Perez. On multiple moving objects. A.I. Memo 883, MIT AI Lab, Boston, MA (USA), May 1986.

[8] Th. Fraichard. Smooth trajectory planning for a car in a structured world. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 318–323, Sacramento, CA (USA), Apr. 1991. IEEE.

[9] K. Fujimura and H. Samet. Motion planning in a dynamic domain. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 324–330, Cincinnatti, OH (USA), may 1990. IEEE.

[10] P. Jacobs, G. Heinzinger, J. Canny, and B. Paden. Planning guaranteed near-time-optimal trajectories for a manipulator in a cluterred workspace. Research Report ESRC 89-20/RAMP 89-15, Engineering Systems Research Center, Univ. of California., Berkeley, CA (USA), Oct. 1989.

[11] K. Kant and S. Zucker. Toward efficient trajectory planning: the path-velocity decomposition. *Int. Journal of Robotics Research*, 5(3):72–89, Fall 1986.

[12] J-C. Latombe. *Robot motion planning.* Kluwer Academic Press, 1990.

[13] C. Ó'Dúnlaing. Motion planning with inertial constraints. *Algorithmica*, 2:431–475, 1987.

[14] J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *Proc. of the Symp. on the Foundations of Computer Science*, pages 144–154, Portland, OR (USA), Oct. 1985. IEEE.

[15] G. Sahar and J. H. Hollerbach. Planning of minimu-time trajectories for robot arms. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 751–758, St Louis, MI (USA), March 1985. IEEE.

[16] C.L. Shih, T.T. Lee, and W.A. Gruver. Motion planning with time-varying polyhedral obstacles based on graph search and mathematical programming. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 331–337, Cincinnatti, OH (USA), May 1990. IEEE.

[17] Z. Shiller and S. Dubowsky. Global time optimal motions of robotic manipulators in the presence of obstacles. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 370–375, Philadelphia, PA (USA), Apr. 1988. IEEE.

[18] K.G. Shin and N.D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Trans. Autom. Contr.*, 30:531–541, June 1985.