Path-Velocity Decomposition Revisited and Applied to Dynamic Trajectory Planning

Article *in* Proceedings - IEEE International Conference on Robotics and Automation · May 1998

DOI: 10.1109/ROBOT.1993.292121 · Source: CiteSeer

2 authors:

Thierry Fraichard
National Institute for Research in Computer Science and Control
150 PUBLICATIONS **4,332** CITATIONS
SEE PROFILE

Christian Laugier
National Institute for Research in Computer Science and Control
334 PUBLICATIONS **6,564** CITATIONS
SEE PROFILE

Proc. of the 1993 IEEE Int. Conf. on Robotics and Automation
May 2-7, 1993. Atlanta, Georgia, USA. Volume 2, pp.40–45.

Path-Velocity Decomposition Revisited and Applied to Dynamic Trajectory Planning[a]

Th. Fraichard and C. Laugier[b]
LIFIA-IRIMAG
46, av. Félix Viallet, 38031 Grenoble Cedex, France
[tf, laugier]@lifia.imag.fr

Abstract — This paper addresses **dynamic trajectory planning**, which is defined as motion planning for a robot $\mathcal{A}$ moving in a **dynamic workspace** $\mathcal{W}$, i.e. a workspace cluttered up with stationary and moving obstacles. Besides $\mathcal{A}$ is subject both to **kinematic constraints**, i.e. constraints involving the configuration parameters of $\mathcal{A}$ and their derivatives, and **dynamic constraints**, i.e. constraints on the forces, the accelerations and the velocities applied to $\mathcal{A}$.

We consider the case of a car-like robot $\mathcal{A}$ with bounded velocity and acceleration, moving in a dynamic workspace $\mathcal{W} = I\!R^2$. Our approach is an extension to the 'path-velocity decomposition' [9]. We introduce the concept of **adjacent paths** and we use it within a novel motion planning schema which operates in two complementary stages: (a) **paths-planning** and (b) **trajectory-planning**. In the paths-planning stage, a set of adjacent paths, one of which leading $\mathcal{A}$ to its goal, are computed. These paths are collision-free with the stationary obstacles and respect $\mathcal{A}$'s kinematic constraints. In the trajectory-planning stage, knowing that $\mathcal{A}$ is able to shift from one path to an adjacent one freely, we determine the motion of $\mathcal{A}$ along and between these paths so as to avoid any collision with the moving obstacles while respecting $\mathcal{A}$'s dynamic constraints.

[a] This work was supported by the French Ministry of Research and Space and the European EUREKA EU-153 project PROMETHEUS Pro-Art.

[b] Research director at INRIA.

Path-Velocity Decomposition Revisited and Applied to Dynamic Trajectory Planning

Th. Fraichard and C. Laugier*
LIFIA-IRIMAG
46, av. Félix Viallet, 38031 Grenoble Cedex, France
[tf, laugier]@lifia.imag.fr

Abstract — This paper addresses **dynamic trajectory planning**, which is defined as motion planning for a robot $\mathcal{A}$ moving in a **dynamic workspace** $\mathcal{W}$, i.e. a workspace cluttered up with stationary and moving obstacles. Besides $\mathcal{A}$ is subject both to **kinematic constraints**, i.e. constraints involving the configuration parameters of $\mathcal{A}$ and their derivatives, and **dynamic constraints**, i.e. constraints on the forces, the accelerations and the velocities applied to $\mathcal{A}$.

We consider the case of a car-like robot $\mathcal{A}$ with bounded velocity and acceleration, moving in a dynamic workspace $\mathcal{W} = I\!R^2$. Our approach is an extension to the 'path-velocity decomposition' [9]. We introduce the concept of **adjacent paths** and we use it within a novel motion planning schema which operates in two complementary stages: (a) **paths-planning** and (b) **trajectory-planning**. In the paths-planning stage, a set of adjacent paths, one of which leading $\mathcal{A}$ to its goal, are computed. These paths are collision-free with the stationary obstacles and respect $\mathcal{A}$'s kinematic constraints. In the trajectory-planning stage, knowing that $\mathcal{A}$ is able to shift from one path to an adjacent one freely, we determine the motion of $\mathcal{A}$ along and between these paths so as to avoid any collision with the moving obstacles while respecting $\mathcal{A}$'s dynamic constraints.

1 Introduction

1.1 Overview of the Problem

An autonomous robot is intended to perform certain actions in its workspace (moving around, grasping and mating parts, etc). Any such action usually implies that a motion is made by the robot. This accounts for the importance of motion planning in Robotics. This importance is reflected in the number of research works which deal with motion planning (the reader is referred to [10] for a recent and quite complete survey of this topic).

Planning a motion for a real robot operating in a real workspace implies being able to cope with such things as *moving obstacles* and the various constraints which restrict the motion capabilities of the robot, e.g. *kinematic constraints*, which involve the configuration parameters of the robot and their derivatives, and *dynamic constraints*, i.e. constraints on the forces, the accelerations and the velocities applied to the robot. As we will see further down, these points have been addressed in the past but, as far as we know, never simultaneously and it is our goal to try to do so.

Thus we will study **dynamic trajectory planning**, which is defined as motion planning for a robot $\mathcal{A}$ moving in a **dynamic workspace** $\mathcal{W}$, i.e. a workspace cluttered up with stationary and moving obstacles. Besides $\mathcal{A}$ is subject to both **kinematic** and **dynamic** constraints.

1.2 Contribution of the Paper

We consider the case of a car-like robot $\mathcal{A}$ with bounded velocity and acceleration, moving in a dynamic planar workspace $\mathcal{W}$. Note that, because it is a wheeled robot, $\mathcal{A}$ is subject to non-holonomic constraints, i.e. non-integrable kinematic constraints which reduce the set of possible differential motions for $\mathcal{A}$ and restrict the geometry of feasible paths accordingly (see §4.1).

Our solution is inspired by the 'path-velocity decomposition' introduced in [9], which addresses motion planning in two complementary stages: (a) planning a geometric path which avoids the stationary obstacles and (b) planning the velocity along this path so as to avoid the moving obstacles. This approach is of a practical interest because it decomposes the original problem into two more simple sub-problems (see §2 for the complexity issues). However it presents a serious drawback: it cannot find a solution if a moving obstacle stops right on the computed path. The usual answer to this problem is to consider a set of candidate paths [12]. Our own answer is the novel concept of **adjacent paths**. Adjacent paths will be formally defined in §5.2.2. Meanwhile, an informal illustration of what adjacent paths are can be found in the roadway: roads are usually divided into several adjacent lanes and every driver knows what passing from one lane to an adjacent one means.

Thanks to this concept, we have designed a novel motion planning schema which also operates in two

*Research director at INRIA.

complementary stages. The first stage, **paths-planning**, takes into account all the time-independent features of the problem at hand (stationary obstacles and $\mathcal{A}$'s kinematic constraints) whereas the second stage, **trajectory-planning**, deals with the time-dependent ones (moving obstacles and $\mathcal{A}$'s dynamic constraints). In the paths-planning stage, a set of adjacent paths, one of which leading $\mathcal{A}$ to its goal, are computed. These paths are collision-free with the stationary obstacles and respect $\mathcal{A}$'s kinematic constraints. In the trajectory-planning stage, knowing that $\mathcal{A}$ is able to shift from one path to an adjacent one freely, we determine the motion of $\mathcal{A}$ along and between these paths so as to avoid any collision with the moving obstacles while respecting $\mathcal{A}$'s dynamic constraints.

The paper is organized as follows: §2 and §3 respectively reviews the complexity issues and the works related to dynamic trajectory planning. Then §4 formally states the problem at hand. §5 describes the solution we have designed in order to solve this problem.

2 Complexity Issues

There are results which suggest that dynamic trajectory planning is generally intractable [18]. Even the two-dimensional case is computationally involved (cf. the NP-hard result established in [2] and the P-Space algorithm presented in [3]). On the other hand, the one-dimensional case seems less intricate (cf. the two polynomial algorithms presented in [11] and [13]) hence the interest of the path-velocity decomposition.

3 Related Works

To our knowledge, [7] and [11] are the only references addressing dynamic trajectory planning, i.e. motion planning with moving obstacles as well as kinematic and dynamic constraints (see §3.3). On the other hand, there are numerous works which are somehow related to our problem. The works addressing either moving obstacles or dynamic constraints are reviewed in the next two sections.

3.1 Moving Obstacles

The general approach which deals with moving obstacles is the 'configuration-time space' approach: it consists in adding the time dimension to the robot's configuration space. Accordingly the different approaches developed in order to solve the path planning problem in the configuration space can be used. Among the various existing works, we can distinguish those based upon extensions of the 'visibility graph' concept [4] and those based upon cell decomposition [15].

3.2 Dynamic Constraints

There are results for exact time-optimal trajectory planning for Cartesian robots subject to bounds on their velocity and acceleration [3, 11]. Besides optimal-control theory provides exact results in the case of robots with full dynamics moving along a given path [1, 17]. 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 a sub-optimal solution. Such grids are defined either in the workspace [16], the configuration space [14], or the state space of the robot [8, 18].

3.3 Dynamic Trajectory Planning

Reference [11] considers the case of a particle with bounded acceleration, moving in a one-dimensional space. Besides it follows and is followed by two moving particles. The solution described operates on both the configuration-time space and the state space of the particle.

The method presented in [7] is more general. It consists in decomposing the configuration-time space of the robot with an octree. Afterwards a search procedure generates a trajectory respecting simple bounds on the velocity, the acceleration and the centrifugal force applied to the robot.

4 Statement of the Problem

4.1 The Robot $\mathcal{A}$

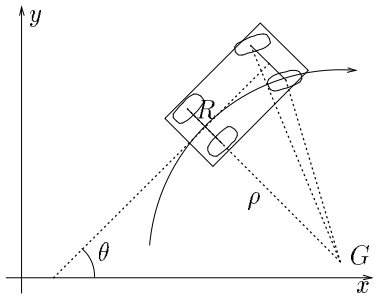

Figure 1: a car-like robot

Let $\mathcal{A}$ be a car-like robot with two rear wheels and two directional front wheels. It is modelled as a polygon moving on the plane $\mathbb{R}^2$. A configuration of $\mathcal{A}$ is completely defined by the 3-tuple $(x, y, \theta) \in \mathbb{R}^2 \times [0, 2\pi[$ where (x, y) are the coordinates of the rear axle midpoint R and θ is $\mathcal{A}$'s orientation (Fig.1).

Assuming pure rolling condition, a wheel can only move in a direction which is normal to its axle. A body moving on the plane has only one centre of rotation.

Therefore, when $\mathcal{A}$ is moving, the axles of its wheels must intersect at this centre of rotation. Let G be $\mathcal{A}$'s center of rotation. G is located on the rear wheels axle (possibly at an infinite distance). The distance ρ between R and G is the curvature radius at point R.

4.1.1 Kinematic Constraints

$\mathcal{A}$ is subject to the two following non-holonomic kinematic constraints (see [10] for more details):

$$\tan\theta = \dot{y}/\dot{x} \quad (1)$$

$$\rho \geq \rho_{min} \quad (2)$$

Equality (1) expresses the fact that R must move in a direction which is normal to the rear wheels axle. Inequality (2) stems from the fact that the front wheels orientation is mechanically limited.

4.1.2 Dynamic Constraints

Let v be the velocity of $\mathcal{A}$ measured along its main axis and let a_{tan} and a_{rad} be respectively the tangential and radial components of $\mathcal{A}$'s acceleration. The following dynamic constraints should be satisfied:

$$|v| \leq v_{max} \quad (3)$$

$$|a_{tan}| \leq a_{max} \quad (4)$$

$$|a_{rad}| \leq g_{max} \quad (5)$$

The meaning of these constraints is quite straightforward. Inequality (3) is a simple speed limit and (4) is an upper bound on the rate of change of speed. The purpose of (5) is to ensure that the radial acceleration does not exceed the counteracting centrifugal acceleration which is supplied by friction between the wheels and the ground.

4.2 The Workspace $\mathcal{W}$

$\mathcal{W} = I\!R^2$, it is cluttered up with stationary obstacles $\mathcal{B}_i^S$, i = $1 \ldots s$, and with moving obstacles $\mathcal{B}_j^M$, j = $1 \ldots m$. Both types of obstacles are modelled as convex polygonal regions of $\mathcal{W}$.

4.3 The Solution

In this framework, a trajectory between an initial configuration q_s and a final configuration q_g is defined by a mapping Γ taking a time $t \in [0, t_f]$ to a configuration $\Gamma(t) = (x(t), y(t), \theta(t))$ and such that $\Gamma(0) = q_s$ and $\Gamma(t_f) = q_g$. The time for the trajectory Γ is t_f. Γ must be **collision-free**, i.e. it must respect:

$$\forall t \in [0, t_f],\ \forall i \in \{1 \ldots s\},\ \forall j \in \{1 \ldots m\},$$
$$\mathcal{A}(t) \cap \mathcal{B}_i^S = \emptyset \text{ and } \mathcal{A}(t) \cap \mathcal{B}_j^M(t) = \emptyset$$

where $X(t)$ designates the region of $\mathcal{W}$ occupied by the object X at time t. Besides Γ must be **feasible**, i.e. it must respect the constraints (1)-(5) presented earlier.

Of course we are interested in finding a **time-optimal** trajectory, i.e. a solution Γ such that t_f should be minimal.

Let us consider the kinematic constraints (1) and (2). $\mathcal{A}$'s path, i.e. the geometric curve followed by R, is a curve in the $xy\theta$-space. However (1) implies that the xy-curve followed by R completely defines this path. Let Υ be this curve: it is the projection of Γ in $I\!R^2$. As a consequence of (1), Υ must be piecewise of class C^1. Besides (2) implies that the curvature of Υ (wherever it is defined) must be upper-bounded by $1/\rho_{min}$ and that the cusp points of Υ should correspond to inversion of $\mathcal{A}$'s direction of motion. Since the concept of adjacent paths becomes quite meaningless when it comes to manœuvering, we will look for a manœuvre-free motion: Υ will be **smooth**, i.e. of class C^1.

5 The Motion Planning Schema

5.1 General Presentation

As mentioned earlier, our approach addresses the problem at hand in two complementary stages. The first stage — **paths-planning** — computes a set of adjacent paths, one of which leading $\mathcal{A}$ to its goal. These paths are collision-free with the stationary obstacles and respect $\mathcal{A}$'s kinematic constraints. Knowing that $\mathcal{A}$ is able to shift from one path to an adjacent one, the second stage — **trajectory-planning** — determines the motion of $\mathcal{A}$ along and between these paths so as to avoid the moving obstacles while respecting $\mathcal{A}$'s dynamic constraints.

5.2 Paths-Planning

Paths-planning is performed in three steps. To begin with, a nominal path Υ_N between q_s and q_g is computed (§5.2.1). Afterwards a set of adjacent paths is automatically derived from Υ_N (§5.2.2). As we will see further down, these adjacent paths are not necessarily collision-free with the stationary obstacles and do not necessarily respect the constraint (2). In order to solve these two problems, parts of the adjacent paths have to be invalidated (§5.2.3).

5.2.1 Planning a Nominal Path

We designed a smooth path planner for a car-like robot. This planner has already been presented in [5] so we will not detail it here. Suffice it to say that it generates a nominal path $\Upsilon_N : [0,1] \longrightarrow I\!R^2$ from q_s to q_g. Υ_N is of class C^1, it is made up of straight segments connected with tangential circular arcs whose radii are greater than ρ_{min}.

5.2.2 Computing Adjacent Paths

A smooth path for $\mathcal{A}$ is a xy-curve Υ of class C^1 and whose curvature (wherever it is defined) is upper-bounded by $1/\rho_{min}$. Assuming that Υ does not intersect itself, a point P located at a distance from Υ smaller than ρ_{min} has a unique normal projection P' on Υ. Let $d_\Upsilon(P)$ be the signed distance[1] between P and its projection on Υ. The path **adjacent to Υ on its left at a distance** δL is defined as: $\Upsilon_l = \{P \in \mathcal{W} \mid d_\Upsilon(P) = \delta L\}$ (Fig.2). Similarly, the path **adjacent to Υ on its right at a distance** δL is defined as: $\Upsilon_r = \{P \in \mathcal{W} \mid d_\Upsilon(P) = -\delta L\}$.

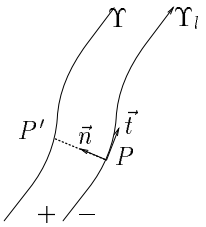

Figure 2: Υ_l, a path adjacent to Υ on its left

Thanks to this definition, it is possible to recursively compute a set of paths adjacent to the nominal path Υ_N on its left and on its right. Let $\mathcal{P} = \{\Upsilon_k, k = 1, \ldots, p\}$ be the whole set of adjacent paths, Υ_N included. Let us denote by $\mathcal{R}(\Upsilon)$ the set of points whose distance to Υ is smaller than $\delta L/2$. The distance δL between any two adjacent paths is chosen so as to ensure that, $\forall \Upsilon \in \mathcal{P}$:

1. The region swept by $\mathcal{A}$ when it follows a path Υ is included in $\mathcal{R}(\Upsilon)$.
2. The region swept by $\mathcal{A}$ on its left (resp. right) side when it leaves a path Υ by making a right (resp. left) turn of radius ρ_{min}, is included in $\mathcal{R}(\Upsilon)$ (Fig.3(a)).
3. The region swept by $\mathcal{A}$ on its left (resp. right) side when it reaches a path Υ by making a right (resp. left) turn of radius ρ_{min} is included, in $\mathcal{R}(\Upsilon)$ (Fig.3(b)).

As we will see further down, these three properties permits to simplify collision checking both along (§5.2.3) and between the paths (§5.3).

5.2.3 Checking Adjacent Paths

The nominal path is collision-free with the stationary obstacles and it respects the kinematic constraints (1) and (2). Unfortunately these two properties do not necessarily hold for an adjacent path. It may happen that such a path is no longer collision-free with the stationary obstacles or does no longer respect the curvature constraint (2). Let $\Upsilon \in \mathcal{P}$: Υ is a mapping $[0,1] \longrightarrow \mathcal{W}$. Collision and curvature checking lead us to compute a set of 'forbidden' intervals in the range $[0,1]$ corresponding to parts of Υ which violate either of these constraints. Let us denote by $\rho(s)$ the curvature radius of the path at the point $\Upsilon(s)$. The set of forbidden intervals for Υ is formally defined as:

$$\begin{aligned}\mathcal{F}(\Upsilon) \quad = \quad & \{[a,b] \subset [0,1] \mid (\forall s \in [a,b],\ \rho(s) < \rho_{min}) \\ & \text{or } (\exists \mathcal{B}_i^S \mid \text{Proj}(\mathcal{B}_i^S, \Upsilon) = [a,b])\}\end{aligned}$$

where $\text{Proj}(\mathcal{B}_i^S, \Upsilon)$ is the 'projection' of the stationary obstacle $\mathcal{B}_i^S$ on the path Υ, i.e. the part of Υ which entails a collision between $\mathcal{A}$ and $\mathcal{B}_i^S$:

$$\begin{aligned}\text{Proj}(\mathcal{B}_i^S, \Upsilon) \quad = \quad & [a,b] \subset [0,1] \mid \forall s \in [a + l_f, b - l_r], \\ & \exists P \in \mathcal{B}_i^S \cap \mathcal{R}(\Upsilon) \mid P' = \Upsilon(s)\end{aligned}$$

where P' is the normal projection of P on Υ and where l_f (resp. l_r) denotes the distance between $\mathcal{A}$'s rear axle and it front-most (resp. rear-most) point. Figure 4 depicts an example of forbidden intervals for a path Υ. The interval $[a_1, b_1]$ corresponds to $\text{Proj}(\mathcal{B}_i^S, \Upsilon)$ while $[a_2, b_2]$ corresponds to a part of Υ which does not respect the curvature constraint (2).

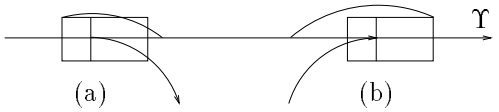

Figure 3: regions swept by $\mathcal{A}$ on its left when leaving/reaching a path Υ by making a right turn

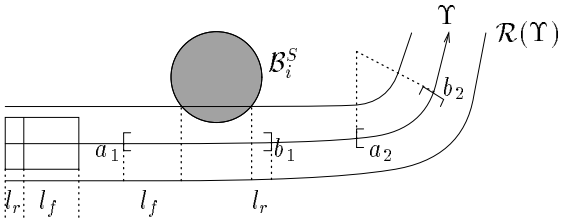

Figure 4: the 'forbidden' intervals of Υ

Note that $\text{Proj}(\mathcal{B}_i^S, \Upsilon)$ is defined by using $\mathcal{R}(\Upsilon)$ instead of the exact region swept out by $\mathcal{A}$ when it moves along Υ. This choice is sound because the region swept out by $\mathcal{A}$ is included in $\mathcal{R}(\Upsilon)$ (cf. prop-

[1] If P is on the left (resp. right) side of Υ, then $d_\Upsilon(P)$ is positive (resp. negative).

erty #1 of §5.2.2). Besides it simplifies the computation of Proj($\mathcal{B}_i^S$,Υ) because $\mathcal{R}(\Upsilon)$ has a much more simple shape.

5.3 Trajectory-Planning

The output of paths-planning is a set $\mathcal{P}$ of adjacent paths and a set $\mathcal{F}$ of forbidden intervals associated with each path. The purpose of trajectory-planning is to determine $\mathcal{A}$'s motion along and between these paths so as to stay out of the forbidden intervals, avoid the moving obstacles and respect the dynamic constraints (3), (4) and (5).

Because of path-changing, i.e. the motion between adjacent paths, the problem at hand is two-dimensional. However it is possible to take advantage of the properties of the adjacent paths in order to reduce the problem to a one-dimensional trajectory planning problem. We have designed a trajectory planner which implements this idea. To begin with, we present a method which determines the trajectory of $\mathcal{A}$ along a given path and then we extend this method so as to incorporate path-changing.

5.3.1 Motion Along a Path

We designed a trajectory planner which is able to compute the trajectory of a given robot $\mathcal{A}$ along a given path Υ so as to avoid moving obstacles and respect dynamic constraints such as (3), (4) and (5). This planner is described in [6]. Classically it reduces the original problem to a one-dimensional problem by parameterizing Υ with a single variable p representing the distance traveled along Υ. Afterwards the dynamic constraints of $\mathcal{A}$ are transformed into constraints on the velocity v and the acceleration a along Υ. The constraints on v are expressed by a velocity limit curve in the state space, i.e. the $p\times v$ plane. On the other hand, the constraints imposed by the moving obstacles can be represented by forbidden regions of the $p\times t$ space, where t represents the time dimension (cf. [9]). In order to deal simultaneously with these two types of constraints, we introduce the novel concept of **state-time space**, i.e. the $p\times v\times t$ space. Let $\mathcal{ST}$ be this state-time space. A curve of $\mathcal{ST}$ represents a trajectory along Υ. Therefore it is possible to solve the problem at hand by searching a curve in $\mathcal{ST}$. The algorithm we designed in order to find such a trajectory operates in the following way: it chooses a time-step τ and assumes that the acceleration applied to $\mathcal{A}$ during a time-step τ is either minimum, null or maximum. Accordingly a state-time has three neighbours and all the state-times that $\mathcal{A}$ can reach from a given state-time lie on a regular grid embedded in $\mathcal{ST}$ (Fig.5).This grid is then searched in order to find a solution. Accordingly trajectory planning is reduced to graph search (cf. [6]).

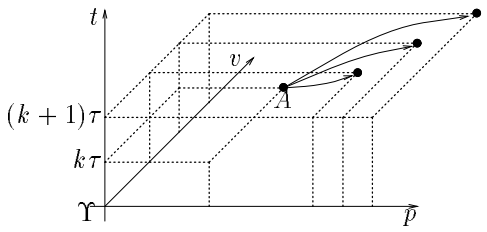

Figure 5: the grid embedded in $\mathcal{ST}$

5.3.2 Motion Between the Paths

Let us consider a path-changing motion as depicted in Fig.6(a). At time t_1, $\mathcal{A}$ shifts smoothly from its current path Υ_k to an adjacent path Υ_{k+1}. $\mathcal{A}$ follows a nominal trajectory Ω and reaches Υ_{k+1} at a certain time t_2.

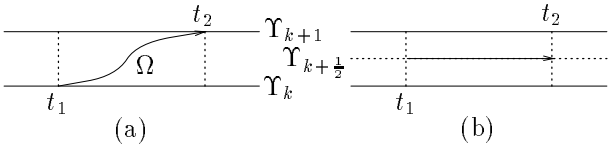

Figure 6: path-changing

Properties #2 and #3 of §5.2.2 ensure that, when $\mathcal{A}$ moves along Ω, it always remains included in the domain $\mathcal{R}(\Upsilon_k)\cup\mathcal{R}(\Upsilon_{k+1})$. Consequently, evaluating whether Ω is collision-free can be done very simply by checking out potential collision in both paths Υ_k and Υ_{k+1} during the time interval $[t_1, t_2]$. Note that, in this case, the obstacles on Υ_k include both the forbidden intervals $\mathcal{F}(\Upsilon_k)$ and the projections of the moving obstacles Proj($\mathcal{B}_j^M(t)$,Υ_k) — such projections being time-dependent.

Accordingly it is possible to model path-changing as a simultaneous motion along Υ_k and Υ_{k+1} during $[t_1, t_2]$, or equally, as a three-step process: (a) at time t_1, $\mathcal{A}$ instantaneously shifts from Υ_k to a fictitious intermediate path $\Upsilon_{k+\frac{1}{2}}$, (b) $\mathcal{A}$ moves along $\Upsilon_{k+\frac{1}{2}}$ during $[t_1, t_2]$ (the obstacles of both Υ_k and Υ_{k+1} are assumed to be projected on $\Upsilon_{k+\frac{1}{2}}$ and (c) at time t_2, $\mathcal{A}$ instantaneously shifts from $\Upsilon_{k+\frac{1}{2}}$ to Υ_{k+1} (Fig.6(b)). Accordingly this modelling reduces path-changing to a one-dimensional motion along a fictitious path.

In this framework. a state-time of $\mathcal{A}$ is a 4-tuple (L, p, v, t) where L is the index of the current path of $\mathcal{A}$. Let us choose a time-step τ and assume that the acceleration applied to $\mathcal{A}$ is either minimum, null or max-

imum. Given a method which determines the path-changing trajectory Ω^2, it is possible to determine the state-time reached by $\mathcal{A}$ at the end of a path-changing. A given state-time has now five neighbours: three on the same path and two on each adjacent paths. All the state-times reachable from a given state-time still lie on a regular grid embedded in $\mathcal{ST}$ and it is still possible to search this grid in order to find a solution.

6 Experimental Results

The algorithm presented above has been implemented in C on a Sun SPARC I. We have tested the algorithm with up to four adjacent paths. In these experiments, the moving obstacles are generated at random without caring whether they collide with each other. An example of trajectory planning involving two paths is depicted in Fig.7. A path is associated with two windows: a trace window showing the part of the grid which has been explored and a result window displaying the final trajectory. Such a window represents the 'time×position' space of the path (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 vertical spacing of the dots corresponds to the time-step τ. In the example, $\mathcal{A}$ starts on the first path (`lane #0`), at position 0 (upper-left corner) with a null velocity. It must reach the end of the first path (right border) with a null velocity. $\mathcal{A}$ can overtake by using the second path (`lane #1`).

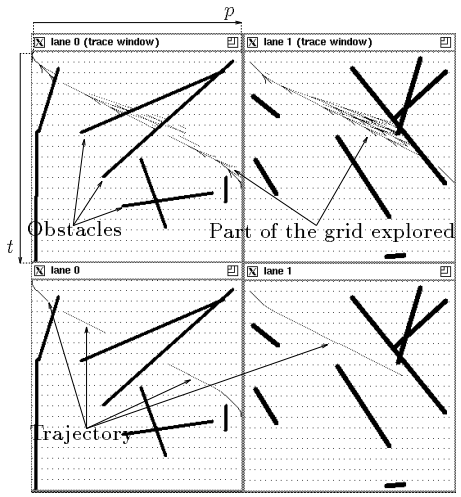

Figure 7: an example of trajectory planning with two paths: the solution has four path-changings

References

[1] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. *Int. Journal of Robotics Research*, 4(3):3–17, Fall 1985.

[2] J. Canny. *The complexity of robot motion planning*. MIT Press, Cambridge, MA (USA), 1988.

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

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

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

[6] Th. Fraichard and C. Laugier. Kinodynamic planning in a structured and time-varying workspace. In C. Laugier, editor, *Geometric Reasoning for Perception and Action*, Lecture Notes in Computer Science. Springer-Verlag, To appear 1993.

[7] K. Fujimura and H. Samet. A hierarchical strategy for path planning among moving obstacles. *IEEE Trans. Robotics and Automation*, 5(1):61–69, Feb. 1989.

[2] Such a method is described in [6] for a car-like robot.

[8] 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.

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

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

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

[12] T-J. Pan and R.C. Luo. Motion planning for mobile robots in a dynamic environment with moving obstacles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 578–583, Cincinnati, OH (USA), May 1990.

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

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

[15] 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 IEEE Int. Conf. on Robotics and Automation*, pages 331–337, Cincinnatti, OH (USA), May 1990.

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

[17] 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.

[18] P.G. Xavier. *Provably-good approximation algorithms for optimal kinodynamic robot motion plans*. PhD thesis, Cornell Univ., Ithaca, NY (USA), april 1992.