

# Partial Motion Planning Framework for Reactive Planning Within Dynamic Environments

Stephane Petti, Thierry Fraichard

▶ **To cite this version:**

Stephane Petti, Thierry Fraichard. Partial Motion Planning Framework for Reactive Planning Within Dynamic Environments. Proc. of the IFAC/AAAI Int. Conf. on Informatics in Control, Automation and Robotics, Sep 2005, Barcelona (SP), France. inria-00182043

**HAL Id: inria-00182043**

**https://inria.hal.science/inria-00182043**

Submitted on 24 Oct 2007

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# PARTIAL MOTION PLANNING FRAMEWORK FOR REACTIVE PLANNING WITHIN DYNAMIC ENVIRONMENTS

Stéphane PETTI
*INRIA Rocquencourt*
*stephane.petti@inria.fr*

Thierry FRAICHARD
*INRIA Rhônes-Alpes*
*thierry.fraichard@inria.fr*

Keywords: Partial Motion Planning, Motion Planning, Navigation, Dynamic Environment, mobile robotics.

Abstract: This paper addresses the problem of motion planning in dynamic environments. As dynamic environments impose a real-time constraint, the planner has a limited time only to compute a motion. Given the intrinsic complexity of motion planning, computing a complete motion to the goal within the time available is, in many real-life situations, impossible to achieve. Partial Motion Planning (PMP) is the answer proposed in this paper to this problem. PMP calculates a motion until the time available is over. At each iteration step, PMP returns the best partial motion to the goal computed so far. Like reactive decision scheme, PMP faces a safety issue: what guarantee is there that the system will never end up in a critical situations yielding an inevitable collision? In this paper the safety issue relies upon the concept of Inevitable Collision States that account for both the system dynamics and the moving obstacles. By computing ICS-free partial motion, the system safety can be guaranteed. Application of PMP to the case of a car-like system in a dynamic environment is presented.

## 1 INTRODUCTION

Motion autonomy has attracted an increasing interest in past years for various robotic systems. Complex systems however exhibit several constraints (kinematic, dynamic, actuators constraints) that restrict their motion capability. These constraints must be accounted for at the motion planning stage in order for the robot to properly execute this plan.

In this paper, we address the problem of robots navigation within dynamic environments through the paradigm of deliberative approaches which consists in planning a priori a trajectory within high dimensional spaces for which it is assumed a model exists. The choice of the state-time space framework is the only suitable to properly express the constraints of the system that can be modelled by a differential equation adding the time dimension to express the moving obstacles present within the environment (Fraichard and Laugier, 1992). Despite the exponentially increasing complexity of motion planning in the dimension space (Canny, 1988), probabilistic planners brought in the mid 90's a new powerful tool for rapid exploration of high dimensional state-time space (Kavraki et al., 1996), (LaValle and Kuffner, 1999). As for the model, the exploration in the time dimension requires a model of the future to be provided. This is admittedly a strong assumption, yet realistic considering latest results on model's prediction (Wang et al., 2003), (Vasquez and Fraichard, 2004).

Further, there are *real time constraints* stemming from a changing environment that have to be considered. At first, the system has the obligation to make a decision within a bounded time, otherwise it might be in danger by the sole fact of being passive. This limited available time for the system to make a decision, *ie.* plan a motion (*decision time constraint*), depends on the nature and dynamicity of the environment. Then, even though a model of the workspace in time is provided, it has a limited time validity only, estimated by the prediction algorithm of these schemes. This *validity duration constraint* requires the motion planner to periodically update its model and calculate a new plan. Even though some work addresses real time motion planning (Brock and Khatib, 2000) a few only consider highly changing environment, performing fast replanning using probabilistic techniques, though for simple systems (Hsu et al., 2002), (Bruce and Veloso, 2002). In our work, we believe that when more complex systems or environments are considered, the real time constraints have to be explicitly considered. In such a case a complete trajectory to the

goal cannot be computed in general and partial plans only can be found. In our work we do not assume an estimate of the world's model can realistically be provided over the complete navigation planning time as assumed in (Feron et al., 2000), but only over limited time period, requiring the planner to periodically calculate completely new plans. The idea of partial planning has already been scarcely mentioned in past and we intend in this paper to settle partial motion planning as a new efficient framework for planning under real time constraints.
Then, when dealing with partial plans, it becomes of the utmost importance to consider the behaviour of the system at the end of the trajectory. What if a car ends its trajectory in front of a wall at high speed? It becomes clear that strong guarantees should be given to this trajectory in order to handle the *safety* issues raised by such a *partial planning*. We use the *Inevitable Collision States* (ICS) formalism recently presented in (Fraichard and Asama, 2004) suitable to establish the relation of the collision states and the dynamic constraints of a system and for which we present the first practical implementation within a motion planning scheme in a dynamic environment.
After briefly presenting past related work in §2, we introduce the notations in §3 that we will carry out for the presentation of the partial motion planner in §4 and the discussion on safety issues in §5. We finally provide in §6 simulation results of an effective navigation within different highly dynamic environments for a car-like robot. We draw our conclusions in §7 over the results and the extensions to be given to this work.

## 2 RELATED WORK

Previous work on motion autonomy largely rely on reactive approaches that explore locally the velocity space of the system from which one admissible control is selected at a time. Motivated first, by a low computational cost and second, by the realistic difficulty to observe and model the environment, these schemes (Borenstein and Koren, 1991), (Khatib, 1996) or (Fiorini and Shiller, 1998) however have two strong limitations : a lack of lookahead, conducting the robot to be trapped in local minima during its trip, and a weak goal directedness keeping the robot from reaching the objective. Furthermore, kinematic or dynamic constraints inherent to car-like robots, addressed by a few specific schemes (Simmons, 1996), (Minguez et al., 2002) or (Fox et al., 1995) remain complex to handle in a general way. Global motion planning schemes have been modified as well in order to gain some reactivity toward changes within the environment. Beside early work based on dynamic programming (Stentz, 1995), the approach of motion planning has mainly benefited from the probabilistic techniques. Only latest work on this issue, recognises the possibility of complete trajectory planning failure and attempts to provide safety guarantee. However the guarantees of $\tau$-safety (Feron et al., 2000) or escape plans (Hsu et al., 2002) do not relate the collision constraint with the dynamics of the system and do not provide necessary guarantees required for safe navigation within highly changing environments.

## 3 NOTATIONS AND DEFINITION

Let $\mathcal{A}$ denote a robotic system placed in a workspace $\mathcal{W}$. The motion model of $\mathcal{A}$ is described by a differential equation of the form $\dot{s} = f(s, u)$ where $s \in \mathcal{S}$ is the state of the system, $\dot{s}$ its time derivative and $u \in \mathcal{U}$ a control. $\mathcal{S}$ is the state space and $\mathcal{U}$ the control space of $\mathcal{A}$. $\mathcal{A}(s)$ is the subset of $\mathcal{W}$ occupied by $\mathcal{A}$ at a state $s$. Let $\phi \in \Phi$: $[t_0, t_f] \longmapsto \mathcal{U}$ denote a control input, *ie.* a time-sequence of controls. Starting from an initial state $s_0$, at time $t_0$, and under the action of a control input $\phi$, the state of the system $\mathcal{A}$ at time $t$ is denoted by $s(t) = \phi(s_0, t)$. An initial state and a control input define a trajectory for $\mathcal{A}$, *ie.* a time sequence of states. The environment is cluttered with a set of obstacles $\mathcal{B}$, closed subset of $\mathcal{W}$. A moving obstacle is time-dependent and denoted by $\mathcal{B}(t)$ and its prediction from time $t_0$ to $\infty$ denoted by $\mathcal{B}(t_0, \infty)$. A state $s$ is a *collision state* at time $t$ if and only if $\exists \mathcal{B}$ such that $\mathcal{A}(s) \cap \mathcal{B}(t) \neq \emptyset$.

Using these notations and definitions, the next section details the Partial Motion Planning (PMP) architecture.

## 4 PARTIAL MOTION PLANNING (PMP) ARCHITECTURE

Planning in a changing environment implies to plan under real time constraints. At first, a robotic system cannot safely remain passive in a dynamic environment as it might be collided by a moving obstacle (*decision constraint*). This decision time $\delta_d$ is function of the environment's dynamicity and could be defined as the minimum time to collision with the obstacles of the environment. Secondly, in a real environment, the model of the future can be predicted over a limited time only $\delta_v$. The planning time (or calculation time $\delta_c$) is hence strictly limited to the minimum of these two bounds. After completion of a planning cycle, it is most likely the planned trajectory of time horizon $\delta_h$ is partial. Thus, the PMP al-

gorithm iterates over a cycle of duration $\delta_c$. We consider in this paper a constant cycle $\delta_c$ in order to be able to regularly get an update of the model, though we can note that in fact, the duration of each cycle does not have to be periodic and should be however of a length $\delta_c$ with $\delta_c = min(\delta_d, \delta_v)$ for the first cycle and $\delta_c = min(\delta_h, \delta_v)$ for the remaining cycles. Let us focus on the planning iteration starting at time $t_i$:

1. An updated model of the future is acquired, *ie.* $\mathcal{B}(t_i, t_i + \delta_v)$ for each moving obstacles.
2. The state-time space of $\mathcal{A}$ is searched using an incremental exploration method that builds a tree rooted at the state $s(t_{i+1})$ with $t_{i+1} = t_i + \delta_c$.
3. At time $t_{i+1}$, the current iteration is over, the best partial trajectory $\phi_i$ in the tree is selected according to given criteria (safety, metric) and is fed to the robot that will execute it from now on. $\phi_i$ is defined over $[t_{i+1}, t_{i+1} + \delta_{h_i}]$ with $\delta_{p_i}$ the trajectory duration.

The algorithm operates until the last state of the planned trajectory reaches a neighbourhood of the goal state. In case the planned trajectory has a duration $\delta_h < \delta_c$, the cycle of PMP can be set to this new lower bound or the navigation (safely) stopped. In practice however, the magnitude of $\delta_h$ is much higher than $\delta_c$.

# 5 SAFETY ISSUES

## 5.1 Definition

Like every method that computes partial motion only (*uncompleteness*), PMP has to face a safety issue: since PMP has no control over the duration of the partial trajectory that is computed, what guarantee do we have that $\mathcal{A}$ will never end up in a critical situations yielding an inevitable collision? We need however to define the safety we consider.

In Fig. 1 we consider a selected milestone of a point mass robot with non zero velocity moving to the right (a state of P is therefore characterised by its position $(x, y)$ and its speed $v$). Depending upon its state there is a region of states (in grey) for which P, even though it is not in collision, will not have the time to brake and avoid the collision with the obstacle. As per (Fraichard and Asama, 2004), it is an Inevitable Collision State (ICS), formally defined as follows:

DEFINITION 1 (INEVITABLE COLLISION STATE)
*A state $s$ is an Inevitable Collision State (ICS) if and only if $\forall\phi$, $\exists t_c$ (for a given $\phi$) such that $\phi(s, t_c)$ is a collision state.*

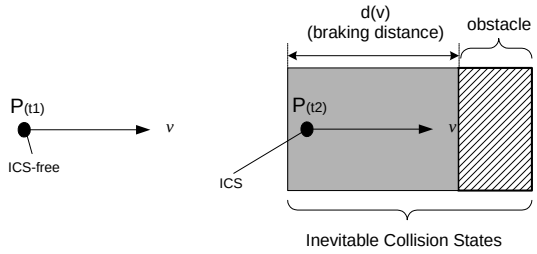



Figure 1: Inevitable Collision State vs. Safe State

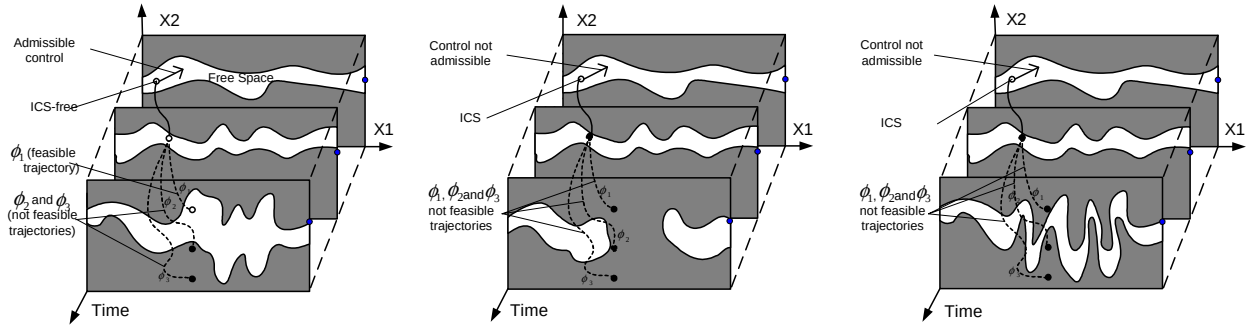
In this paper, we refer to a safe state as an ICS-free state.

## 5.2 Safe State Calculation

In general, computing the ICS for a given system is an intricate problem since it requires to consider the set of all the possible future trajectories. To compute in practice the ICS for a system such as $\mathcal{A}$, it is taken advantage of the approximation property established in (Fraichard and Asama, 2004). This property shows that a conservative approximation of the ICS can be obtained by considering only a finite subset $\mathcal{I}$ of the whole set of possible future trajectories. Figure 2 illustrates this property. In Fig. 2(a), a safe trajectory is easily found within an expansive free space, therefore the state checked is safe. In Fig. 2(b), the space is obstructed and a safe trajectory cannot be found. The state is thus considered as not sage. In Fig. 2(c) an admissible safe trajectory certainly exists but is not found over the given set of three trajectories within such a space with low expansiveness. The state is then conservatively considered as not safe. In an extremely dynamic environment it would be similar, thus there are no necessary bounds on the environment dynamics, as long as it is fully observable and that a model of the future is provided. In the PMP algorithm, every new state is similarly checked to be an ICS or not over $\mathcal{I}$. In case all trajectories appear to be in collision in the future, this state is an ICS and is not selected.

## 5.3 Safe Partial Trajectories

A safe trajectory consists of safe states. However, a practical problem appears when safety has to be checked for the continuous sequence of states defining the trajectory. In order to solve this problem and further reduce the complexity of the PMP algorithm, we present a property that simplifies the safety checking for a trajectory. To begin with, we demonstrate a property that states that for a given control input, all the states between an ICS and the corresponding collision state, are ICS.




(a) ICS-free state (b) ICS in an obstructed environment (c) ICS in a highly dynamic world

Figure 2: Inevitable Collision State calculation over a subset of trajectories $\mathcal{I} = \{\phi_1, \phi_2, \phi_3\}$ for $\mathcal{A}$

PROPERTY 1
*Let $s$ be an ICS at time $t_0$. For a given control input $\phi$, let $t_c$ denote the time at which a collision occurs. Then $\forall t \in [t_0, t_c], s(t) = \phi(s, t)$ is also an ICS.*

*Proof: suppose that a state $s_i = \phi(s, t_i)$, with $t_i \in [0, t_c]$, is not an ICS. By definition, $\exists \phi^j$ that yields no collision when applied to $s_i$. Let $\phi^i$ denote the part of $\phi$ defined over $[t_0, t_i]$. Clearly, the combination of $\phi^i$ and $\phi^j$ also yields no collision when applied to $s \rightsquigarrow$ contradiction.* $\diamond$

We can now provide a sufficient safety condition for a partial trajectory that states that provided a trajectory is collision free, if the last state of the trajectory is not an inevitable collision state then none of the states of the trajectory are inevitable collision states.

PROPERTY 2 (SUFFICIENT SAFETY CONDITION)
*Given a trajectory defined over $[t_0, t_f]$, if*
(H1) *the trajectory is collision-free and*
(H2) *$s(t_f)$ is not an ICS*
*then $\forall t \in [t_0, t_f]$, $s(t)$ is not an ICS.*

*Proof: Suppose that $\exists t_i \in [t_0, t_f]$ such that $s_i = s(t_i)$ is an ICS. Then, by definition, $\forall \phi, \exists t_c$ such that $\phi(s_i, t_c)$ is a collision state. If $t_0 \leq t_c \leq t_f$ then collision occurs before $t_f \rightsquigarrow$ contradiction with H1. Now, if $t_c > t_f$ then by previous property P1, we must have $s(t_f)$ is an ICS $\rightsquigarrow$ contradiction with H2.* $\diamond$

This property is important since first, it proves a trajectory is continuously safe while the states safety is verified discretely only, and second it permits a practical computation of safe trajectories by integrating a dynamic collision detection module within existing incremental exploration algorithms, like A* or Rapidly-Exploring Random Tree (RRT) (LaValle and Kuffner, 1999).

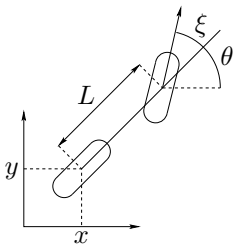



Figure 3: The car-like vehicle $\mathcal{A}$ (bicycle model).

# 6 CASE STUDY: NAVIGATION OF A CAR-LIKE ROBOT

## 6.1 Vehicle Model

We consider the dynamic system of a car-like robot $\mathcal{A}$. A state of $\mathcal{A}$ is defined by the 5-tuple $s = (x, y, \theta, v, \xi)$ where $(x, y)$ are the coordinates of the rear wheel, $\theta$ is the main orientation of $\mathcal{A}$, $v$ is the linear velocity of the rear wheel, and $\xi$ is the orientation of the front wheels (Fig. 3). A control of $\mathcal{A}$ is defined by the couple $(\alpha, \gamma)$ where $\alpha$ is the rear wheel linear acceleration. and $\gamma$ the steering velocity. The motion of $\mathcal{A}$ is governed by the following differential equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_r \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} v_r \cos\theta \\ v_r \sin\theta \\ \frac{\tan\xi v_r}{L} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \alpha + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \gamma \qquad (1)$$

with $\alpha \in [\alpha_{min}, \alpha_{max}]$ (acceleration bounds), $\gamma \in [\gamma_{min}, \gamma_{max}]$ (velocity steering bounds), and $|\xi| \leq \xi_{max}$ (steering angle bounds). $L$ is the wheelbase of $\mathcal{A}$.

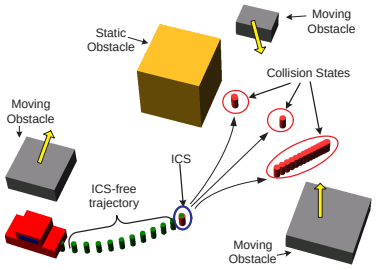



Figure 4: safe state calculation

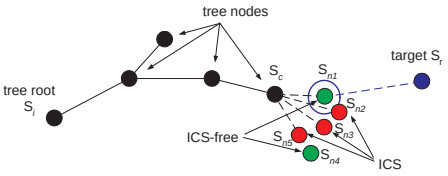



Figure 5: Tree contruction over a PMP cycle

## 6.2 ICS Calculation

For our application we consider the subset $\mathcal{I}$ of braking trajectories obtained by applying respectively constant controls $(\alpha_{min}, \dot{\xi}_{max})$, $(\alpha_{min}, 0)$, $(\alpha_{min}, \dot{\xi}_{min})$ until the system has stopped. Once the system is still, it is checked to be collision free (*ie.* over a trajectory obtained by applying constant (0,0) controls) until the end of the PMP cycle. With this respect, a $\tau$ safe state as introduced in (Feron et al., 2000) can be considered as an ICS-free state during $\tau$ seconds over this latter subset. Fig. 4 illustrates how a state of the partial trajectory is checked to be an ICS or not. The collision states pointed by the arrows represent the collision that will occur in the future from this state for all braking trajectories of $\mathcal{I}$. In this case, since all trajectories collide in the future, this state is an ICS.

## 6.3 Tree Construction

In our work, we used the efficient RRT method and replaced the geometric collision checker with our inevitable collision state checker. The control space of our system is reduced to the set of bang bang controls $\tilde{\mathcal{U}}$=$(\alpha, \dot{\xi})$ with $\alpha \in [\alpha_{min}, 0, \alpha_{max}]$ and $\dot{\xi} \in \left[\dot{\xi}_{max}, 0, \dot{\xi}_{min}\right]$

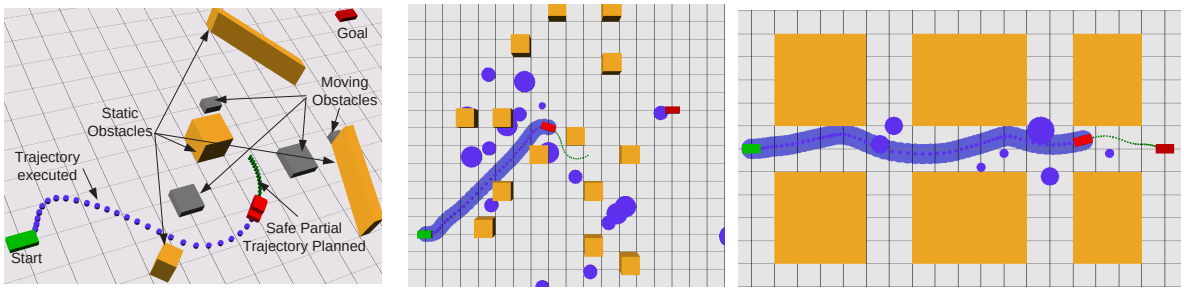
The exploration of the state-time space consists in building incrementally a tree as follows (Fig. 5): a milestone $s_r$ is generated within $\mathcal{W}$ with a probability $p$ to be the goal. The closest state $s_c$ to $s_r$ is selected. A control from $\tilde{\mathcal{U}}$ is applied to the system during a fixed time (integration step). In case the new state $s_n$ of the system is not an ICS, this control is valid. The operation is repeated over all control inputs and finally the new state, safe and closest to $s_r$, is finally selected and added to the tree.

## 6.4 Results

The first implementation of the PMP in various dynamic environment proves the efficiency of the approach as a navigation function for a car-like robot. The software is implemented in C++ and run on a Pentium4@1.7GHz. The parameters of the PMP are a cycle $\delta_c$ of 1s, an integration step of 0.5s and for the car $v_{max} = 2.0m/s$, $\xi_{max} = \pi/3rad$, $\dot{\xi}_{max} = 0.2rad/s$, $\alpha_{max} = 0.1m/s^2$. The environment is supposed fully observable with a validity satisfying the PMP constraints (*ie.* $\delta_v > \delta_c + \delta_b$). At the initial state (left) the car is still and ICS-free, and the goal state (right) is at still as well. Fig. 6(a) illustrates an example of a navigation within an environment cluttered by moving and static obstacles. One can observe a result of a the safe partial trajectory (ahead of the vehicle) during a PMP cycle that avoids the obstacles. Behind the car, the trace is the trajectory, built from partial trajectories from all previous PMP cycles and (ideally) executed by the robot. We observe how the car is obliged to slow down at the intersection of several obstacles, since no other safe trajectories could be found, before to re-accelerate. Fig. 6(b) illustrates how the algorithm provides efficient navigation to the car evolving within a highly dynamic environment (20 circle moving obstacles). Finally, Fig. 6(c) illustrates a case study of such an implementation within a city where the vehicle has to avoid pedestrians (circle obstacles). Videos can be found at `http://emotion.inrialpes.fr/fraichard/pmp-films`.

# 7 CONCLUSION

In this paper we tackle the problem of navigation for a car-like robot within a dynamic environment and propose a Partial Motion Planning scheme (PMP) which handles the *real-time constraint* inherent to such an environment, while accounting for the kinematic and dynamic constraints of the system. The PMP algorithm consists in iteratively exploring the state-time space during a fixed limited time and building a tree using incremental techniques. During a cycle, a complete trajectory calculation to the goal can not be guaranteed in general, which raises the issue of the *safety* of our system. We use the formalism of the *Inevitable collision States* (ICS) as the theoretical answer to this safety problem. We demonstrate a property that allows a discrete verification of safe




(a) safe Partial Motion Planning (b) Navigation within a highly dynamic environment (c) Navigation on a one-way pedestrian-friendly road

Figure 6: results of safe motion plans

states while guaranteeing the safety over the whole trajectory and allows its first practical implementation in a planning scheme within a dynamic environment. Thus, we present an original and effective algorithm navigating safely a car-like robot within highly dynamic environment. Finally, simulation results prove the overall effectiveness of the complete PMP algorithm and show a case-study for a real implementation within a pedestrian city area. Future work includes the coupling of the PMP algorithm with a closed loop control and its integration on an experimental vehicle. Our goal is to perform experimentations within a real environment, for which a model of the future obstacles' behaviour will be determined thanks to a prediction technique.

## REFERENCES


Borenstein, J. and Koren, Y. (1991). The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288.

Brock, O. and Khatib, O. (2000). Real time replanning in high-dimensional configuration spaces using sets of homotopic paths. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, San Francisco (US).

Bruce, J. and Veloso, M. (2002). Real-time randomized path planning for robot navigation. In *Int. Conf. on Intelligent Robots and Systems*, Lausanne, Switzerland.

Canny, J. (1988). *The complexity of Robot Motion Planning*. MIT Press, Cambridge, MA.

Feron, E., Frazzoli, E., and Dahleh, M. (2000). Real-time motion planning for agile autonomous vehicles. In *AIAA Conference on Guidance, Navigation and Control*, Denver (US).

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772.

Fox, D., Burgard, W., and Thrun, S. (1995). The dynamic window approach to collision avoidance. Technical Report IAI-TR-95-13.

Fraichard, T. and Asama, H. (2004). Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024.

Fraichard, T. and Laugier, C. (1992). Kinodynamic planning in a structured and time-varying 2D workspace. In *Int. Conf. on Robotics and Automation*, Nice, (FR).

Hsu, D., Kindel, R., Latombe, J.-C., and Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. *Int. Journal of Robotics Research*, 21(3):233–255.

Kavraki, L., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12:566–580.

Khatib, M. (1996). *Sensor-based motion control for mobile robots*. PhD thesis, LAAS-CNRS December, 1996.

LaValle, S. and Kuffner, J. (1999). Randomized kinodynamic planning. In *Int. Conf. on Robotics and Automation*, pages 473–479, Detroit (US).

Minguez, J., Montano, L., and Santos-Victor, J. (2002). Reactive navigation for non-holonomic robots using the ego kinematic space. In *Int. Conf. on Robotics and Automation*, Washington (US).

Simmons, R. (1996). The curvature velocity method for local obstacle avoidance. In *International Conference on Robotics and Automation*, pages 3375–3382, Minneapolis (USA).

Stentz, A. (1995). The focussed D* algorithm for real-time replanning. In *Int. Joint Conf. on Artificial Intelligence*, pages 1652–1659, Montreal, Quebec.

Vasquez, D. and Fraichard, T. (2004). Motion prediction for moving objects: a statistical approach. In *Int. Conf. on Robotics and Automation*, New Orleans, LA.

Wang, C.-C., Thorpe, C., and Thrun, S. (2003). Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan.