Submitted to the Advanced Robotics journal

Visibility-based probabilistic roadmaps for motion planning

T. Siméon, J-P. Laumond, and C. Nissoux

LAAS-CNRS
7, avenue du Colonel-Roche
31077 Toulouse Cedex - France
{nic,jpl,nissoux}@laas.fr

Abstract

This paper presents a variant of Probabilistic Roadmap Methods (PRM) that recently appeared as a promising approach to motion planning. We exploit a free-space structuring of the configuration space into visibility domains in order to produce small roadmaps, called visibility roadmaps. Our algorithm integrates an original termination condition related to the volume of the free space covered by the roadmap. The planner has been implemented within a software platform allowing to address a large class of mechanical systems. Experiments show the efficiency of the approach in particular for capturing narrow passages of collision-free configuration spaces.

1 Introduction

Due to the continuous increasing power of the computers, probabilistic approaches to motion planning [1] (e.g.,[2, 8, 18, 10, 3]) allow today to solve practical problems which were not addressed few years ago. Apart some attempts aiming to provide formal models of complexity [13, 9, 6, 12], the success of such methods remains better noticed than well understood.

This paper proposes a variant of the Probabilistic RoadMap (PRM) algorithm introduced in [8] (and independently in [18] as the Probabilistic Path Planner). These algorithms generate collision-free configurations randomly and try to link them with a simple local path planning method. A roadmap is then generated, tending to capture the connectivity of the collision-free configuration space CS_{free}.

Our variant of these approaches takes advantage of the visibility notion. While usually each collision-free configuration generated by the PRM algorithm is integrated to the roadmap, our algorithm keeps only configurations which either connect two connected components of the roadmap, or are not "visible" by some so-called

[1] See [14] for a comprehensive introduction to motion planning

guard configurations. This approach computes roadmaps with small number of nodes. It integrates a termination condition related to the volume of the free space covered by the roadmap. Experimental comparison shows good performances in terms of computation time, especially when applied to configuration spaces with narrow passages.

Section 2 introduces the notion of visibility roadmap. Section 3 describes a simple probabilistic algorithm that computes such roadmaps. The algorithm is analyzed in Section 4 and discussed in Section 5 relatively to the other PRM based algorithms proposed in the literature. Finally, the last section presents several examples of problems solved by the planner that is integrated into the `Move3d` software platform dedicated to motion planning.

2 Visibility Roadmaps

Consider a mechanical system moving in a workspace defined by a set of obstacles. Let CS denotes the configuration space of the system and CS_{free} be the free space defined by the open subset of collision-free configurations with respect to the obstacles.

Local methods Given two configurations q and q' of the mechanical system, a *local method* refers to an algorithm that computes an admissible path $\mathcal{L}(q, q')$ connecting both configurations in the absence of obstacle. The notion of admissible path is related to the feasibility with respect to the kinematic constraints on the motions. Figure 12 illustrates several examples of local methods: $\mathcal{L}$ may simply correspond to a straight-line path when the motions are not constrained, to Reeds&Shepp curves [19] in the case of non holonomic mobile robots, or to Manhattan paths for mechanical systems requiring to move only one degree of freedom at a time.

Roadmaps A *roadmap* is a graph whose nodes are collision-free configurations. Two nodes q and q' are adjacent if the path $\mathcal{L}(q, q')$ computed by the local method lies in CS_{free}. Roadmaps are used to solve motion planning problems by the so-called *query* procedure: given two configurations q_{init} and q_{goal}, the procedure first connects q_{init} (resp. q_{goal}) to the roadmap R if there exists q^*_{init} (resp. q^*_{goal}) such that $\mathcal{L}(q_{init}, q^*_{init}) \subset CS_{free}$ and $\mathcal{L}(q_{goal}, q^*_{goal}) \subset CS_{free}$. Then the procedure searches for a path in the extended roadmap. If such a path exists, the solution of the motion planning problem appears as a path constituted by a finite connected sequence of subpaths computed by $\mathcal{L}$.

Visibility domains For a given local method $\mathcal{L}$, the visibility domain of a configuration q is defined as the domain:

$$Vis_{\mathcal{L}}(q) = \{q' \in CS_{free} \text{ such that } \mathcal{L}(q, q') \subset CS_{free}\}$$

Configuration q is said to be the *guard* of $Vis_{\mathcal{L}}(q)$.

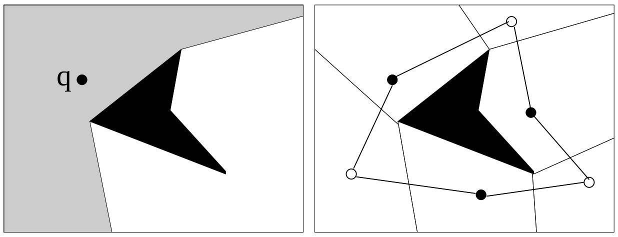


Figure 1: Visibility domain of a configuration and the visibility roadmap defined by three guards nodes (black) and three connection nodes (white). Here paths $\mathcal{L}(q, q')$ are the straight-line segments $[q, q']$.

Free-space Coverage A set of guards constitutes a *coverage* of CS_{free} if the union of their visibility domains covers the free space CS_{free}. Note that the existence of finite coverage both depends on the shape of CS_{free} and on the local method $\mathcal{L}$. Such finite coverage may not always exist. This issue is related to the notion of ϵ-*goodness* introduced in [9].

Visibility Roadmaps Consider now s visibility domains $Vis_{\mathcal{L}}(q_i)$ such that the s guards do not "see" mutually through the local method, i.e., $\mathcal{L}(q_i, q_j) \not\subset CS_{free}$ for any pair of guards (q_i, q_j). Then we build the following graph R. Guards $\{q_i\}_{i=1,s}$ are nodes of the graph. For any two intersecting visibility domains $Vis_{\mathcal{L}}(q_i)$ and $Vis_{\mathcal{L}}(q_j)$, we add a node q, called a *connection node*, and two edges (q, q_i) and (q, q_j) (see Figure 1). The graph R is said to be a *visibility roadmap.* R clearly verifies the following property:

Property: Let us assume a visibility roadmap R *whose set of guards covers* CS_{free}*. Let us consider any two configurations* q_{init} *and* q_{goal} *such as there exists a connected sequence of collision-free paths of type* $\mathcal{L}$ *between them. Then there is a guard node* q_1 *and a guard node* q_2 *in* R *such as:* $q_{init} \in Vis_{\mathcal{L}}(q_1)$, $q_{goal} \in Vis_{\mathcal{L}}(q_2)$ *with* q_1 *and* q_2 *lying in a same connected component of* R.

The notion of visibility roadmap raises several comments:

- Since the definition of the visibility domains is related to a local method, it would have been better to use the term of "reachable domain". Both notions are identical when the local method simply computes straight line segments. We keep the word "visibility" because it is more intuitive.
- We consider implicitly that R is an undirected graph: that means that $\mathcal{L}$ is assumed to be symmetric.
- Finally the number of guards is not required to be optimal. Optimality refers to the well known and challenging art gallery problem [16].

3 A probabilistic algorithm

Principle The algorithm that we propose below is general. It allows to build visibility roadmaps without requiring any explicit computation of the visibility domains.

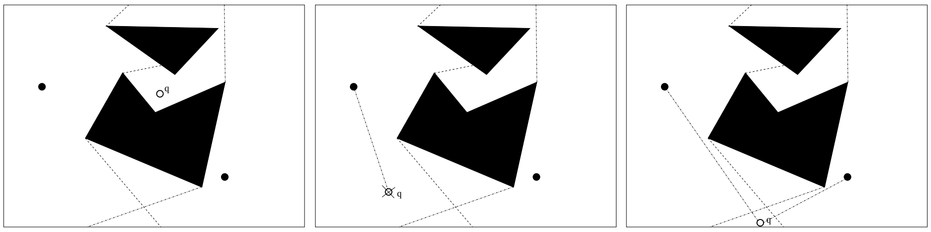


Figure 2: The three cases: a random-free configuration is **a/** a new guard inserted to the roadmap, **b/** rejected, or **c/**a connection node merging two connected components.

The roadmap is constructed incrementally by randomly sampling the configuration space and attempting to connect some pairs of collision-free samples by the local method. Figure 2 illustrates the principle of the sampling strategy used at each iteration of the algorithm. Randomly chosen configurations are checked for collision to generate samples in CS_{free}; when a free sample is found, it is added to the roadmap either if it does not "see" any another node of the current roadmap (ie. it is a new guard), or if it is seen by at least two nodes belonging to two distinct connected components of the roadmap (i.e. it is a connection node). The end of the roadmap's construction is controlled by a termination condition related to the volume of free space currently covered by the roadmap.

Description The algorithm, called `Visib-PRM`, iteratively processes two sets of nodes: *Guard* and *Connection*. The nodes of *Guard* belonging to a same connected component (i.e. connected by nodes of *Connection*) are gathered in subsets G_i.

Algorithm `Visib-PRM`

$Guard \leftarrow \emptyset$; $Connection \leftarrow \emptyset$; $ntry \leftarrow 0$
While $(ntry < M)$
  Select a random free configuration q
  $g_{vis} \leftarrow \emptyset; G_{vis} \leftarrow \emptyset$
  For all components G_i of $Guard$ **do**
    $found \leftarrow FALSE$
    For all nodes g of G_i **do**
      If(q belongs to $Vis(g)$) **then**
        $found \leftarrow TRUE$
        If $(g_{vis} = \emptyset)$ **then** $g_{vis} \leftarrow g$; $G_{vis} \leftarrow G_i$
        Else /* *q is a connection node* */
          Add q to $Connection$
          Create edges (q, g) and (q, g_{vis})
          Merge components G_{vis} and G_i;
    until $found = TRUE$
    If $(g_{vis} = \emptyset)$ **then**/* *q is a guard node* /*
      Add $\{q\}$ to $Guard$; $ntry \leftarrow 0$
    Else $ntry \leftarrow ntry + 1$
End

At each elementary iteration, the algorithm randomly selects a collision-free configuration q. The main loop processes all the current components G_i of $Guard$. The algorithm loops over the nodes g in G_i, until it finds a node visible from q. The first time the algorithm succeeds in finding such a visible node g, it memorizes both g and its component G_i and switches to the next component G_{i+1}. When q "sees" another guard g' in another component G_j, the algorithm adds q to the $Connection$ set and the component G_j is merged with the memorized G_i. If q is not visible from any component, it is added to the $Guard$ set. The main loop fails to create a new node when q is visible from only one component; in that case q is rejected.

Parameter $ntry$ is the number of failures before the insertion of a new guard node. $1/ntry$ gives an estimation of the volume not yet covered by visibility domains. It estimates the fraction between the non covered volume and the total volume of CS_{free}. This is a critical parameter which controls the end of the algorithm. Hence, the algorithm stops when $ntry$ becomes greater than a user set value M, which means that the volume of the free space covered by visibility domains becomes probably greater than $(1 - \frac{1}{M})$.

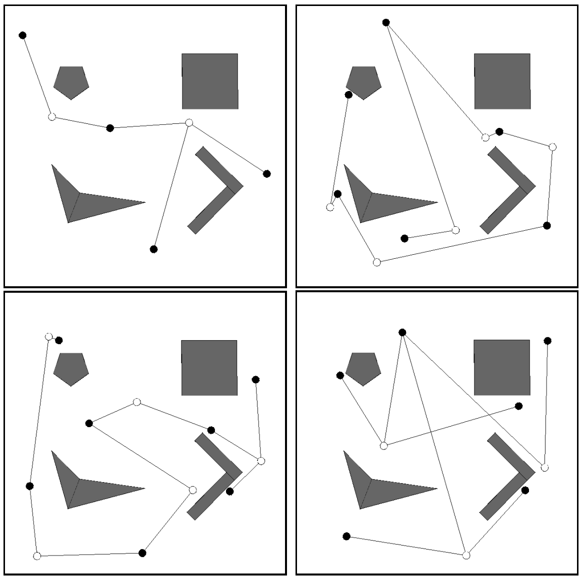


Figure 3: Four runs of the algorithm on a simple example with $M = 50$.

4 Analysis

Size of visibility roadmaps The roadmaps computed by `Visib-PRM` capture the free-space connectivity with a small number of guards. This is illustrated onto the two-dimensionnal example of Figure 3 that shows visibility roadmaps computed by several runs of the algorithm: the size of the roadmaps (i.e. the number of guards) varied over the different runs between four (optimal coverage) and seven guards. All four roadmaps, generated with M set to 50, completely cover the free space; their size would therefore remain the same if the algorithm was run with a larger value of M. This illustrates an interesting property of the algorithm: the size of the produced roadmaps, although not optimal, remains intrinsic to the complexity of the CS_{free}. It is bounded by the maximal number of guards that cover the free space without mutual visibility.

More complicated problems may obviously require more than a few guards to provide a coverage of free-space. Their number remains however reasonably small compared to the difficulty of the problem: for example, less than one hundred guards are sufficient to solve the 6-dof problem of Figure 4 (see Section 5 for more details). The solution path shown onto the figure was found from a visibility roadmap computed by the algorithm with $M = 400$.

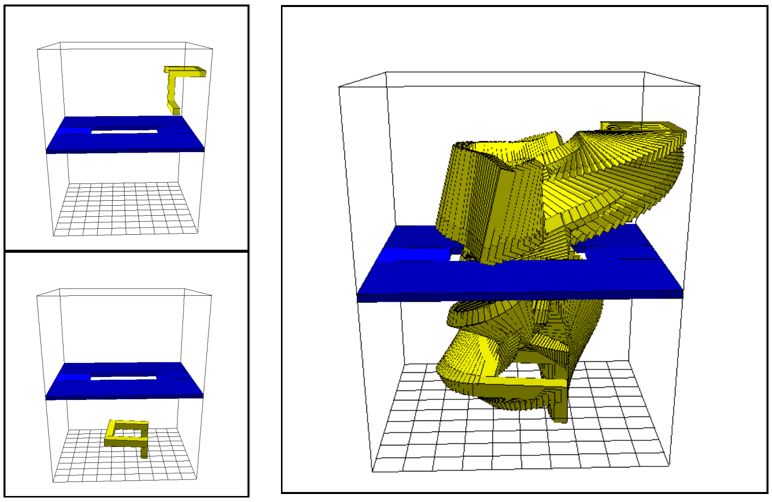


Figure 4: A solution path for the 6-dof puzzle problem presented in [11].

Termination criterion Each new guard node inserted to the roadmap increases the coverage of CS_{free}. Therefore the probability of generating configurations in non covered regions keeps decreasing over the iterations. The algorithm is then guaranteed to terminate for any value of M. When it stops, a probabilistic estimation of the percentage of free space not covered by the guards is $\frac{1}{M}$. This also means that path-planning queries may succeed to connect configurations to the roadmap with a probability of $(1 - \frac{1}{M})$.

This estimation clearly tends towards the real coverage when M tends toward infinity. Experiments performed onto several examples show a rapid convergence of the estimated coverage: Figure 5 shows the evolution of the free-space coverage for the 6-dof puzzle problem of Figure 4, with respect to the number of guards generated by the algorithm. The solid curve plots the coverage estimated by $(1 - \frac{1}{ntry})$. The dashed monotonic curve plots the real percentages of the free space coverage (they were computed by testing the visibility of each set of guards with a huge number of free configurations). As shown by the figure, the estimation rapidly converges toward the actual coverage: the error becomes less than one percent with 100 guards. Also one can note that despite the rather complex shape of CS_{free} for this example, less than hundred guards are sufficient to see 98% of the free space.

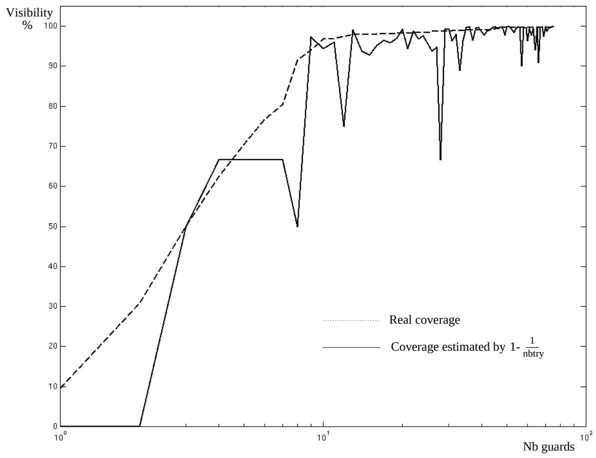


Figure 5: Free space coverage w.r.t. the number of guards for the 6-dof puzzle problem

Side-effect of the algorithm The random generation of the guards may produces in some cases guards that will be difficult to connect. This effect is illustrated in Figure 6 where two guards have been generated near the boundary of the black triangular obstacle. They fully cover CS_{free}, however the intersection of both visibility domains is "unfortunately" small.

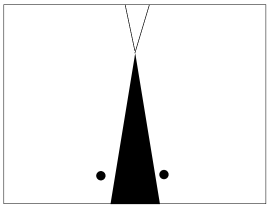


Figure 6: A pathologic case built by the algorithm

The only way to complete the roadmap is to pick a connection node in the small triangle. Then the algorithm will fail if the parameter M is not sufficiently high. Nevertheless this case is only a side-effect of the algorithm. Indeed, in this example, the probability to select the first two guards with a small intersection domain is very low [17]. Moreover, this undesirable effect was never observed in practice onto all the examples we experimented with the algorithm.

5 Related work and comparison

`Visib-PRM` was inspired by the success of probabilistic roadmap techniques (PRM) introduced in in [8, 18] as a way of solving practical planning problems in possibly high dimensional configurations spaces. This section first surveys several PRM-based planners proposed in the literature. It also provides some elements of comparaison with our algorithm and finally presents experiments showing the good performance of **Visib-PRM** in the presence of narrow passages compared to a simplified version of these planners.

The PRM scheme Various planning methods using a probabilistic roadmap scheme are described in several recent papers [1, 4, 6, 5, 7, 8, 10, 11, 18]. The underlying concept shared by these planners is to first construct a roadmap of simple paths connecting collision-free configurations picked at random, and then to use this roadmap to answer multiple path-planning queries. The various algorithms proposed for the roadmap construction, mostly differ by the sampling strategy used to generate a new node and the attempts made to connect this node to the roadmap.

The simpliest version of the algorithm, called **Basic-PRM** in several papers (e.g. see [10]) , creates nodes with all configurations picked uniformly at random in CS_{free}, checks the connections with all the nodes of the roadmap, and stops after a given number of nodes is reached. Such simple strategy generally requires dense roadmaps to capture the free space connectivity and more sophisticated strategies have been proposed to overcome this problem: by adding an enhancement stage that further samples the difficult regions [11], by creating a greater density of nodes near the boundary of the free-space [1, 4], by allowing random reflexions onto CS-obstacles [5], or by defining a multi-stage strategy based on a dilatation of the free-space [7].

Most of them were proposed and shown to improve the efficiency of the roadmap construction when the free-space contains narrow passages, difficult to capture with uniform sampling strategies. They may however require more complicated geometric operations (e.g. distance penetration [7]) or may be sensitive to some parameter tuning to work well in particular environments. The effect of narrow passages onto the complexity of **Basic-PRM** was also investigated in several other papers based onto some properties of the free-space called $\epsilon - goodness$ [9], expansiveness [6] or path clearance [9].

Comparison With respect to these methods, the main advantage of the visibility roadmap is its natural small size, intrinsic to the complexity of the problem. This

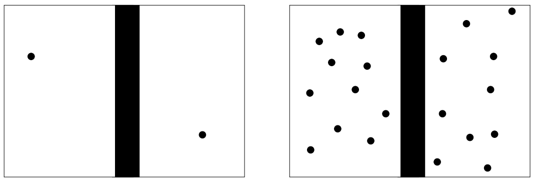


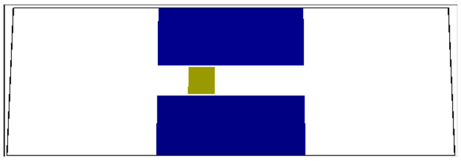
Figure 7: Visibility roadmap versus classical PRM

advantage is illustrated in Figure 7. In this example CS_{free} has two connected component. Our algorithm will quickly find the two guards that cover both connected components, and the algorithm will stop just after M elementary loops. At each of these loops the local method will be called only *twice*, while a **Basic-PRM** algorithm should call the local method a lot of times to check whether each new collision-free node can be connected to the roadmap. With a same number n of random collision-free configurations, the visibility roadmap algorithm will call the local method $O(n)$

times, while `Basic-PRM` will call it $O(n^2)$ times[2].

Concerning the performance of the algorithm, the two main steps of any PRM-based algorithm are the sampling strategy used to generate a new node and the tests required to connect this node to the current roadmap. Both steps require expensive geometric operations for checking whether a configuration or a path found by the local method are collision-free. The sampling strategy used by `Visib-PRM` is more expensive than the uniform sampling generally performed by other PRM algorithms (instead of adding any collision-free configuration to the roadmap, `Visib-PRM` may reject several free samples before a new guard is found). It is however important to note than testing whether a configuration is collision-free is *far less* expensive than checking the connections to the roadmap. Hence, this step requires repetitive calls to the collision checker along all the paths generated by the local method, which completely dominates the work when the size of the roadmap becomes large. Because of the small size of visibility roadmaps, this suggests that the extra cost paid for the more selective sampling strategy of `Visib-PRM` is largely compensated by a notable gain of performance to establish the connections to the roadmap. In particular, the experiments provided in the next section show a good overall performance of visibility roadmaps in presence of narrow passages that usually require a high density of nodes when using uniform sampling.

Experimental comparison onto narrow passages Consider the two canonical examples illustrated by Figures 8 and 9. The free space consists of two unit squared regions connected by a narrow passage of width $\epsilon << 1$. For each example, we compared for different values of ϵ the number l of calls to the local planner required to capture the connectivity of the free space. The values of l directly reflect the performance of both algorithms in term of CPU times since the geometric operations (collision checking) required for testing the validity of a local path is the most time consuming step of the algorithm.

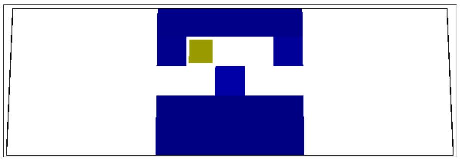


	Basic-PRM		Visib-PRM		*gain*
$\frac{1}{\epsilon}$	$l_b(*1000)$	$\#nodes$	$l_v(*1000)$	$\#nodes$	$\frac{l_b}{l_v}$
100	33	363	14	5	2.3
1.000	2.508	3252	132	5	19
10.000	269.667	35987	1577	5	171

Figure 8: a first canonical example of narrow passages

The tables summarize for each example, the results obtained for narrow passages of increasing difficulty, after averaging the values of l over several runs of each algorithm.

[2] Limiting the calls to a given neighborhood (see [11] for instance) may allow to reach a linear complexity with a "well" chosen value of the threshold distance; however, a larger size n may be



$\frac{1}{\epsilon}$	Basic-PRM $l_b(*1000)$	$\#nodes$	Visib-PRM $l_v(*1000)$	$\#nodes$	*gain* $\frac{l_b}{l_v}$
12	18	250	4	13	4.5
25	142	635	8	13	17
50	2.000	2.900	30	13	66
100	31.000	11.700	145	13	213

Figure 9: a second canonical example of narrow passages

- **Roadmap's size:** With Basic-PRM the number of roadmap's nodes increases linearly (resp. quadratically) with $\frac{1}{\epsilon}$ for the first (resp. second) example. Figures 10-abc show the roadmaps computed by Basic-PRM for increasing values of $\frac{1}{\epsilon} = 12, 25, 50$. Vis-PRM produces for both examples small roadmaps whose size remains independant of ϵ. For the first example, the visibility roadmap only contains 5 nodes: three guard nodes (one in each large free-space components and one in the narrow passage) linked by three connection nodes. For the other example, the created roadmap shown in Figure 10-d only contains 13 nodes (seven guard nodes linked by six connection nodes).

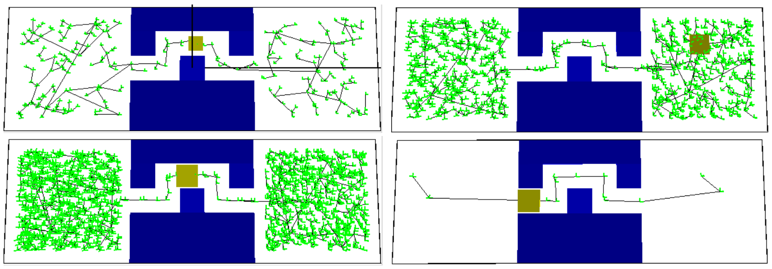
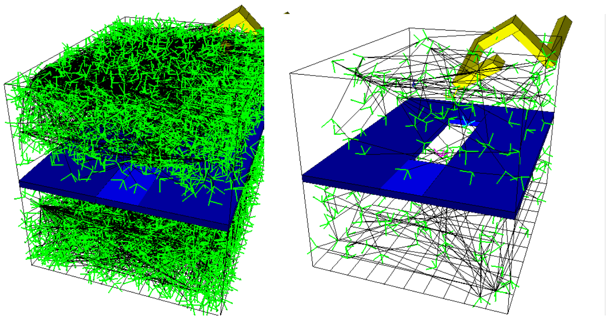


Figure 10: Roadmaps produced by Basic-PRM for $\frac{1}{\epsilon} = 12$ (left-up), $\frac{1}{\epsilon} = 25$ (down-left), $\frac{1}{\epsilon} = 50$ (down-left) and by Visib-PRM for any value of ϵ (down-right)

- **Performance:** The relative performance of both methods for capturing the narrow passage is expressed by the gain $g = \frac{l_b}{l_v}$ between their respective numbers of calls to the local method. The performance of the visibility algorithm over the basic algorithm increases with the difficulty of the problem: g respectively increases in $O(\epsilon)$ and $O(\epsilon^2)$ for the examples of figures 8 and 9.

Figure 11 summarizes the results of another experiment performed onto a 6-dof "puzzle" example [11]. The free-space of this example consists of two large regions

required to maintain the roadmap's connectedness.



PRM	Basic	Visibility
Roadmap size	4723	103
CS_{free} coverage	99.9%	99.7%
Random confs	14169	14369
Free confs	4723	4753
Local method #calls	700610	57622
Col. checker #calls	8.725985	1.121790
CPU time	3367 sec	281 sec

Figure 11: Comparison onto the 6-dof puzzle example.

connected by a narrow passage. Here, the complexity of the narrow passage clearly depends on the relative width of the hole with respect to the smallest dimension of the moving object, and also on the total volume of the configuration space. The results summarized in the table were obtained for the following dimensions: the workspace has dimension $200 * 150 * 150$ with a rectangular passage of width 30 and the moving object is made up five blocks of length 50 and cross-section 10. Each test was performed by running both algorithms with the *same set* of configurations (i.e. the random node generator was initialized with the same seed) until the roadmap captured the connection between the two large regions. All the values of the table were averaged over ten runs performed with different seeds. `Visib-PRM` captured the free space connectivity with a roadmap much smaller than `Basic-PRM` while maintaining the same coverage of the free space. Moreover, the performance (number of calls to the local method or CPU time) was 12 times better. Notice that both required roughly the same total number of random configurations.

Similar experiments performed onto this example tend to show that the gain still increases for larger sizes of the workspace (it drops to 18 when doubling the volume of the free-space) or a smaller width of the passage. On the other side, comparisons performed onto the less constrained examples of Section 6 showed a gain of performance ranging from one to a few times. For all cases, the produced visibility roadmaps are much smaller.

6 Experiments

Both algorithms were integrated into the `Move3d` software platform[3] that we developped at LAAS. `Move3d` allows to model a large class of mechanical systems: free-flying, manipulators, nonholonomic mobile vehicles, mobile manipulators… The library contains a set of functionalities that make easy the development of new motion planners. Collision detection was performed with the *I-Collide* library [15].

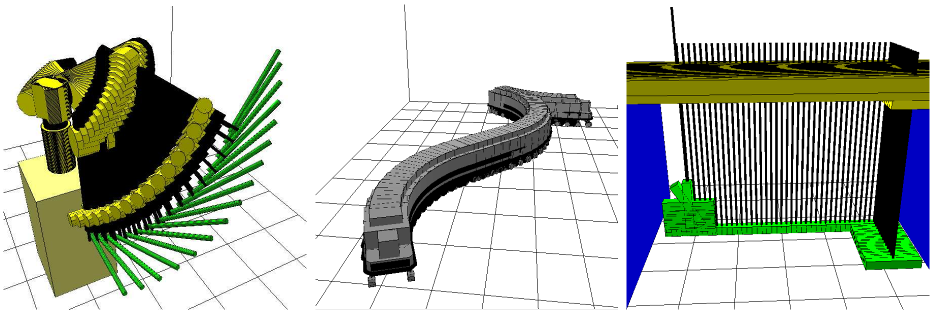


Figure 12: Local methods used for the examples : straight-line segment, Reeds&Shepp paths and Manhattan paths

Figures 13, 14, 15 and 16 present various problems solved by `Visib-PRM` for several kinds of mechanical systems: a robot arm, a mobile manipulator, a rolling bridge and an articulated hand. The upper part of each figure shows the environment and the visibility roadmap computed to solve the planning problem illustrated by the lower part: the initial and final configurations are shown on the left and the trace of the computed path is displayed on the right. The corresponding local methods illustrated by Figure 12 are respectively:

- Robot arm: straight line segments
- Mobile manipulator: combination of Reeds&Shepp paths [19] for the non holonomic platform and straight line segments for the manipulator
- Rolling bridge: Manhattan paths (in this example the kinematic constraint imposes to move one degree of freedom at a time).
- Articulated hand: straight line segments

It is important to note that all the examples were solved by the *same* planner based onto the `Visib-PRM` algorithm presented in the paper. Here, the algorithm includes start and goal configurations as initial guards and stops as soon as it finds a solution path. The following table summarizes the performance of the planner for solving the first problem. Other queries re-using the computed roadmaps would require less than a second for the four examples.

Mechanical system	dof	Local method	Roadmap's size	CPU time
Robot arm	6	Linear	26	370 sec.
Mobile manipulator	9	Linear/ReedsShepp	75	55 sec.
Rolling bridge	4	Manhattan	25	2 sec.
Articulated hand	25	Linear	70	90 sec.

[3] see `http://www.laas.fr/~nic/Move3D`

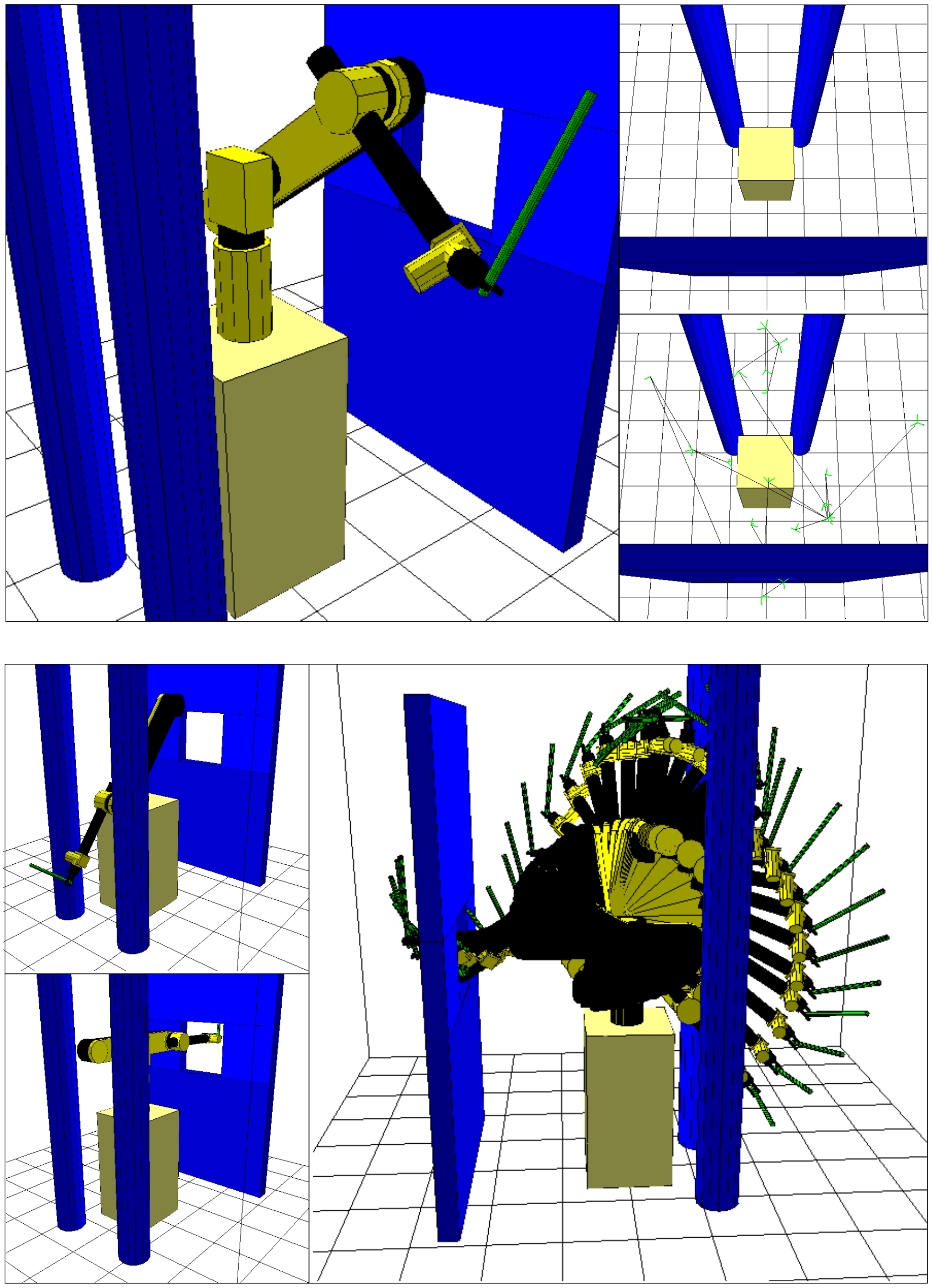


Figure 13: A 6-dof robot arm

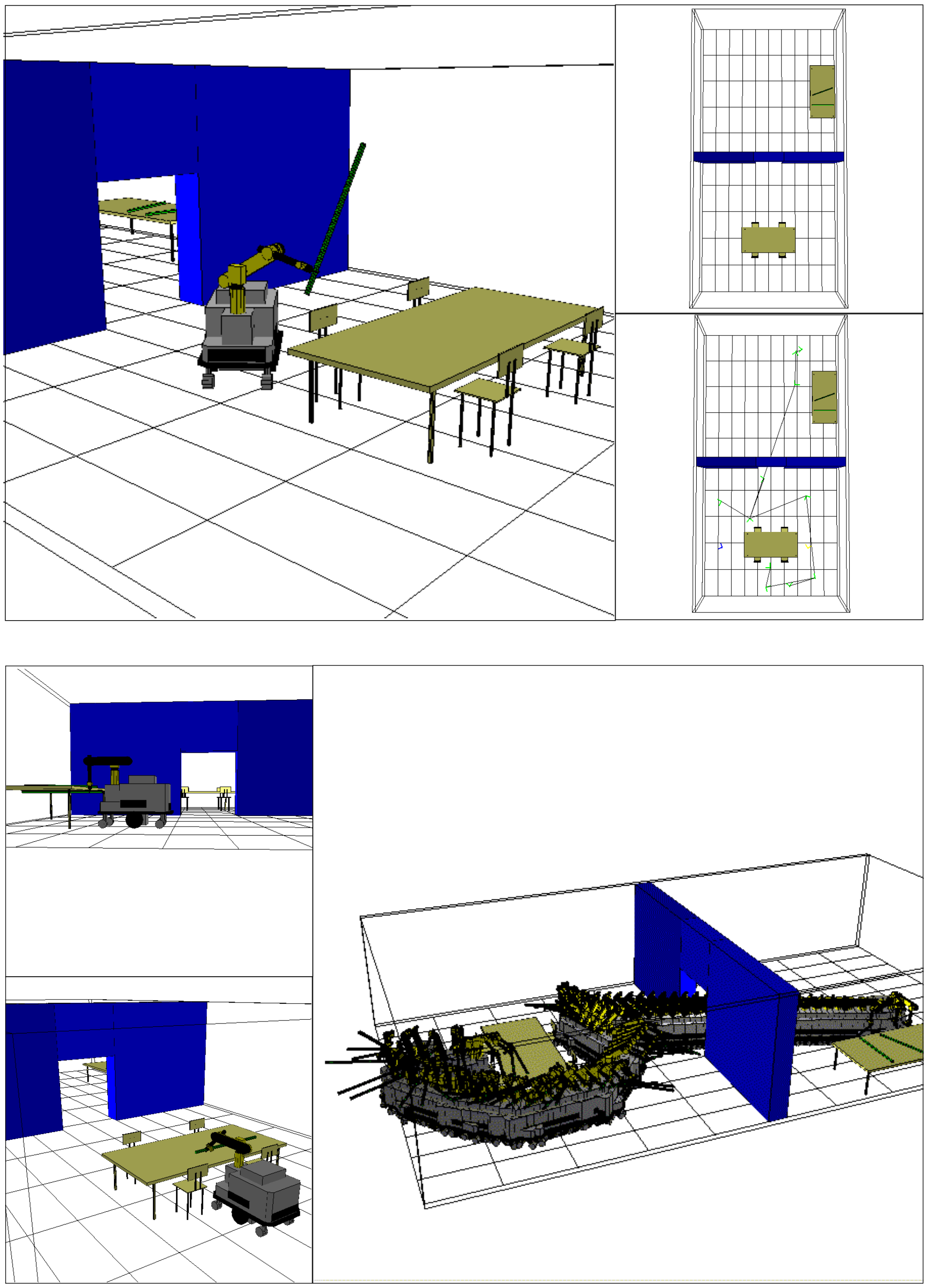


Figure 14: A 9-dof mobile manipulator (non holonomic)

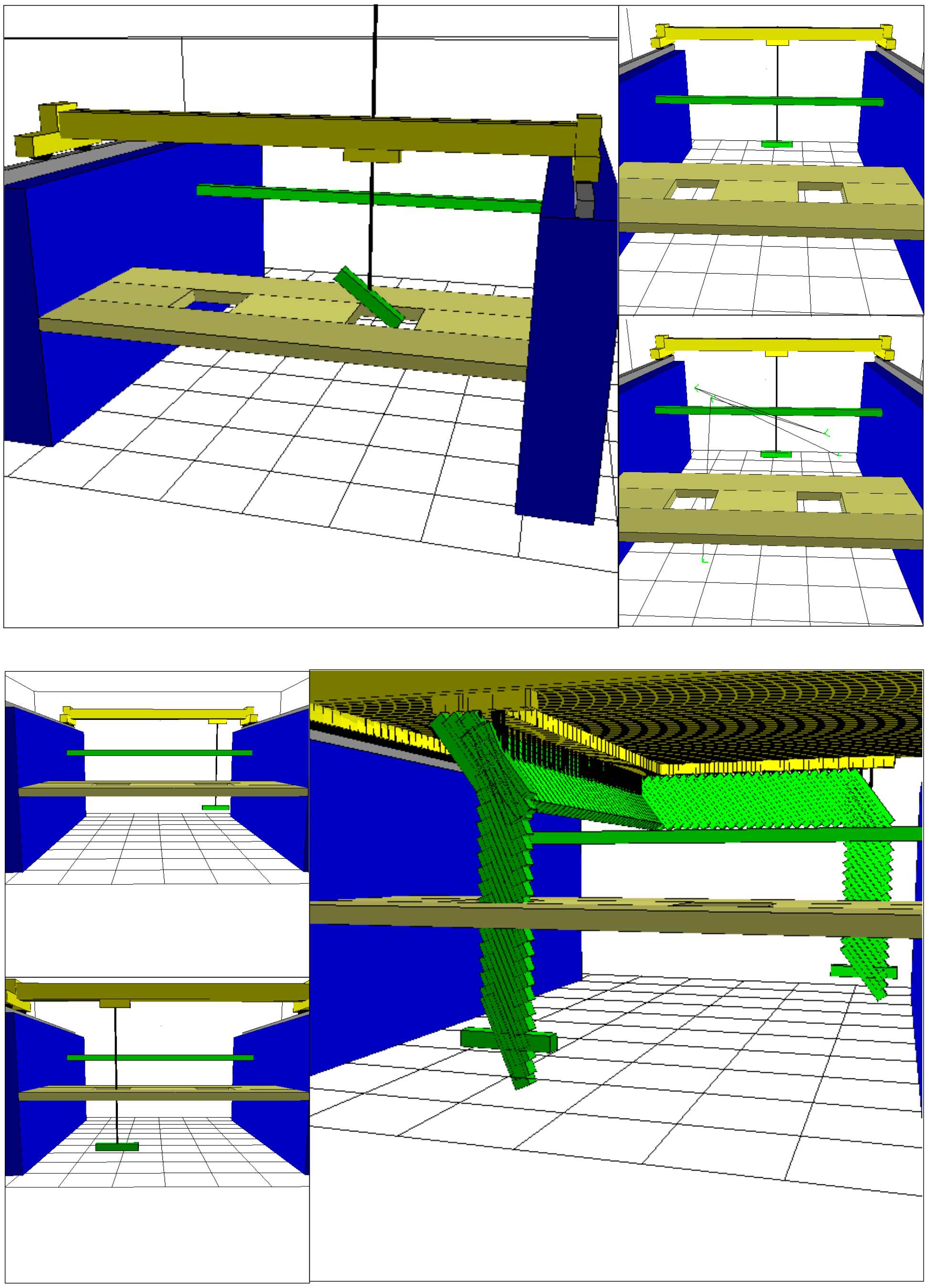


Figure 15: A 4-dof rolling bridge

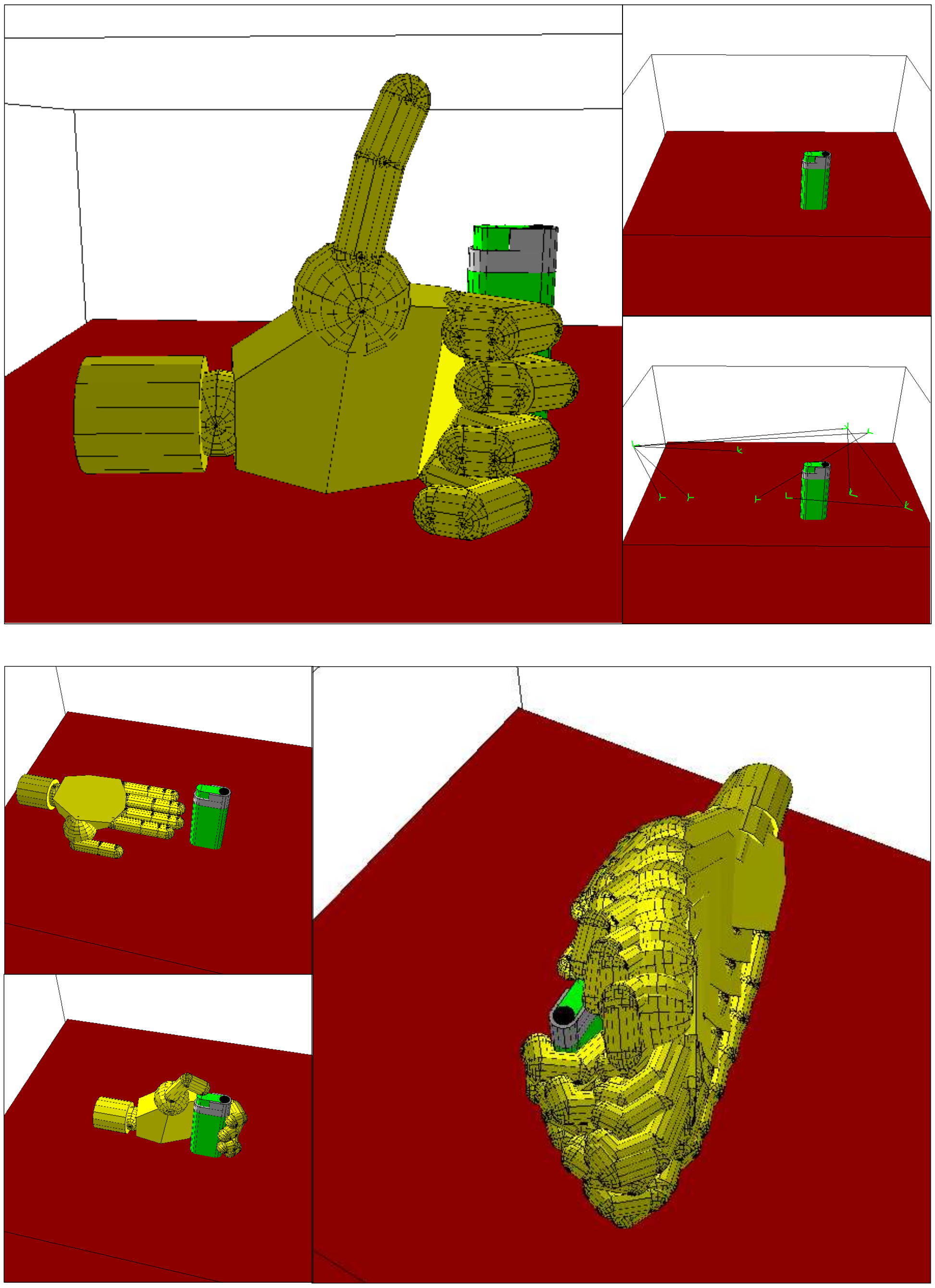


Figure 16: A 25-dof articulated hand

7 Conclusion

In this paper we have introduced the notion of visibility roadmaps as a way to capture the free-space connectivity with a small number of guards and also, to control the end of the roadmap's computation by estimating the quality of the free-space coverage within the algorithm. The first results obtained with the algorithm are encouraging, especially for the difficult cases of free-spaces containing narrow passages which usually require dense and expensive roadmaps with other PRM planners.

These results raise several interesting issues that we expect to address. It may be interesting to investigate the idea of "movable guards" in order to limit the risk of generating guards whose visibility domains have a very small intersection. Also, a formal analysis of the proposed method would be desirable to better characterize its performance from geometric properties of the free-space.

References

[1] N. Amato, O Bayazit, L. Dale, C. Jones and D. Vallejo. OBPRM: an obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective (WAFR98)*, P. Agarwal and all (Eds), AK Peters 1998.

[2] J. Barraquand and JC. Latombe. Robot motion planning: a distributed representation approach. In *International Journal of Robotics Research*, vol 10 n 6, 1991.

[3] P. Bessiere, J. Ahuactzin, T, El-Ghazali and E. Mazer. The "Ariane's clew" algorithm: Global planning with local methods. In *IEEE International Conference on Robots and Systems*, 1993.

[4] V. Boor, M. Overmars and A. Frank van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *IEEE International Conference on Robotics and Automation*, Detroit (USA), 1999.

[5] T. Horsh, F. Schwartz and H. Holle. Motion planning for many degrees of freedom - Random reflexions at CSpace obstacles. In *IEEE International Conference on Robotics and Automation*, San Diego (USA), 1994.

[6] D. Hsu, JC. Latombe, R. Motwani. Path planning in expansive configuration spaces. In *IEEE International Conference on Robotics and Automation*, Albuquerque (USA), 1997.

[7] D. Hsu, L. Kavraki, JC. Latombe, R. Motwani and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Robotics: The Algorithmic Perspective (WAFR98)*, P. Agarwal et al (Eds), AK Peters, 1998.

[8] L. Kavraki and JC Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE International Conference on Robotics and Automation*, San Diego (USA), 1994.

[9] L. Kavraki, L. Kolountzakis and JC. Latombe. Analysis of probabilistic roadmaps for path planning. In *IEEE International Conference on Robotics and Automation*, Minneapolis (USA), 1996.

[10] L. Kavraki, P. Švestka, J.-C. Latombe and M.H. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces, In *IEEE Transactions on Robotics and Automation vol 12, n4, 1996.*

[11] L. Kavraki and JC. Latombe. Probabilistic roadmaps for robot path planning. In *Practical Motion Planning in Robotics*, K. Gupta and A. del Pobil (Eds), Wiley Press, 1998.

[12] L. Kavraki, M. Kolountzakis and J.-C. Latombe. Analysis of Probabilistic Roadmaps for Path Planning, In *IEEE Transactions on Robotics and Automation vol 14, n1, 1998.*

[13] F. Lamiraux and JP. Laumond. On the expected complexity of random path planning. In *IEEE International Conference on Robotics and Automation*, Minneapolis (USA), 1996.

[14] J.C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[15] M. Lin, D. Manocha, J. Cohen and S. Gottschalk. Collision detection: Algorithms and applications. In *Algorithms for Robotic Motion and Manipulation (WAFR96)*, JP. Laumond and M. Overmars (Eds), AK Peters 1997.

[16] J. Goodman and J. O'Rourke. *Handbook of Discrete and Computational Geometry.* CRC Press, 1997.

[17] C. Nissoux. Visibilité et méthodes probabilistes pour la planification de mouvement en robotique. *PdD Thesis (in french), University Paul Sabatier, Toulouse, 1999.*

[18] M. Overmars and P. Svestka. A Probabilistic learning approach to motion planning. In *Algorithmic Foundations of Robotics (WAFR94)*, K. Goldberg et al (Eds), AK Peters, 1995.

[19] J. A. Reeds and R. A. Shepp. Optimal paths for a car that goes both forward and backwards. *Pacific Journal of Mathematics*, 145 (2), pp. 367–393, 1990.