Notes on Visibility Roadmaps and Path Planning

J.-P. Laumond, *LAAS-CNRS, Toulouse, France*
T. Siméon, *LAAS-CNRS, Toulouse, France*

Abstract

This paper overviews the probabilistic roadmap approaches to path planning whose surprising practical performances attract today an increasing interest. We first comment on the configuration space topology induced by the methods used to steer a mechanical system. Topology induces the combinatorial complexity of the roadmaps tending to capture both coverage and connectivity of the collision-free space. The we introduce the notion of optimal coverage and we provide a probabilistic scheme in order to compute what we called visibility roadmaps [26]. Reading notes conclude on recent results tending to better understand the behavior of these probabilistic path planning algorithms.

1 Introduction

The framework of this work lies in the tentative to provide path planners working for large classes of mechanical systems. Such a generality is imposed by an increasing number of path planning applications that extend the robotics area where the research has been initially conducted [19].

In our case, we are interested in providing CAD systems with path planning facilities in the context of logistics and operation in industrial installations[1]. A typical scenario is the following one. An operator has to define a maintenance operation involving the moving of an heavy freight. He has a CAD software including all the geometric details and facilities to display the plant together with catalogue containing lists of available tools to perform the maintenance task. He should choose suitable handling devices among cranes, rolling bridges, carts... and validate his choice by simulating the task within the CAD system. As the other facilities offered by the CAD software, the use of path planners should be as easy as possible. The operator does not care about path planning algorithms.

Probabilistic path planners, and among them, probabilistic roadmap algorithms appeared as a promising route to face the constraints imposed by such a scenario.

This paper does not contain new algorithms, new theorems nor new original results. It constitutes a set of remarks and working notes on probabilistic roadmaps algorithms introduced at the beginning of the 90's [13, 25, 15] and now attracting numerous research groups, including ours.

The first section starts from the beginning: it deals with the controllability of mechanical systems independently from any computational perspective. In the next section we put emphasis on the choice of the methods used to steer a mechanical system from a configuration to another one. The set of configurations reachable from a given configuration has various shapes depending on the considered steering method. Paving the space with reachable sets ask for combinatorial topology issues which are discussed. The roadmaps induced by such pavements capture both coverage and connectivity of the space and allow to apply a retraction approach to path planning. At this stage, we introduce the notions of optimal coverage and visibility roadmaps. Section 4 addresses the computational point of view: probabilistic algorithms are today the only effective way to pave the space with reachable sets. After presenting the principle of the probabilistic

[1]This work is supported by the European Esprit Project 28226 MOLOG (http://www.laas.fr/molog).

roadmap algorithms we show how visibility roadmaps may be computed. An interest of the algorithm lies in its control. Another one is the small size of the computed roadmaps. The work related to probabilistic roadmap algorithms is reported and commented in Section 5.

2 Controllability and CS topology

Examples of mechanical systems Consider the mechanical systems displayed in Figure 1 together with examples of collision-free paths. Modeling a mechanical system in the context of path planning addresses two issues.

The first one deals with the placement constraints. We should identify the configuration space CS of the system, i.e. a minimal set of parameters locating the system in its workspace. This is an easy task for the robot arm, the mobile robot and the rolling bridge, all of them being open kinematic chains. The system constituted by the two holonomic mobile robots manipulating an dumbbell is a closed kinematic chain. The placement parameters of both robots are linked by equations modeling the grasping of the dumbbell and giving rise to holonomic constraints. For this special example it is possible to select five independent parameters defining the configuration space properly, the other placement parameters being deduced by explicit equations involving only the five independent parameters. For more general closed loop chains, characterizing CS properly is not an easy task.

Second, we should consider the kinematic constraints. There is no special constraint for the robot arm, any configuration parameter being a degree of freedom directly controlled by a motor independently from any other one. Any path in CS is admissible. The same property holds for the coordinated path of the two holonomic mobile robots. The control of the rolling bridge may impose special constraints, like moving a degree of freedom at once. In that case the only admissible paths in CS are Manhattan paths. The motion of the mobile manipulator is submitted to the constraint of rolling without sliding. This is a nonholonomic constraint that affects the range of admissible paths, but not the dimension of the reachable configuration space.

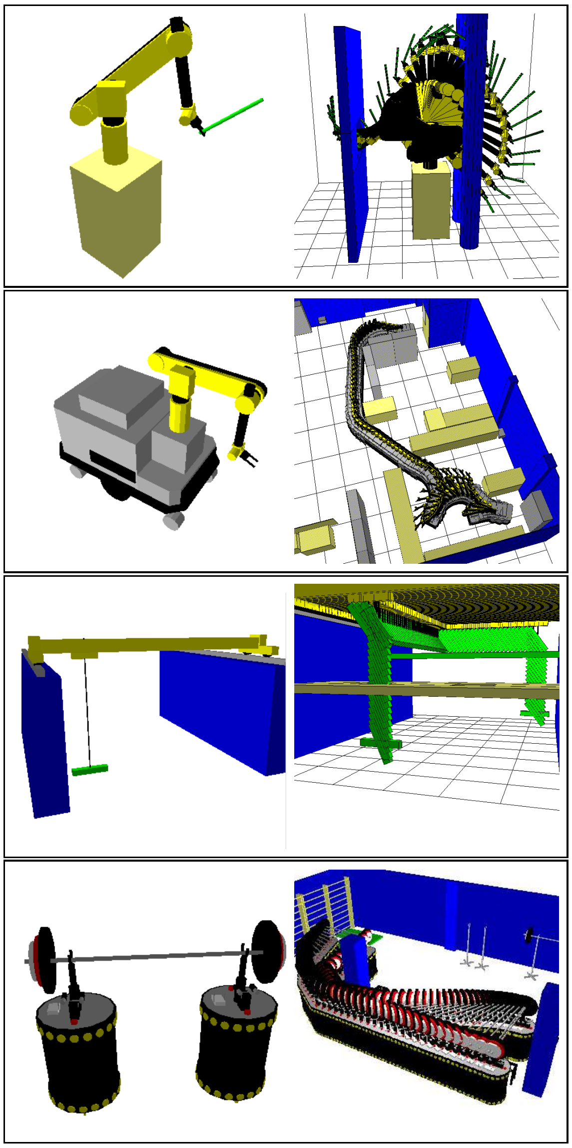


Figure 1: *Examples of mechanical systems and collision-free paths.*

Controllability All the previous systems are small-time controllable: starting from any configuration, the set of configurations reachable before any given time contains a neighborhood of the starting configuration.

Let us translate this property in geometric terms: the set of configurations reachable by all the admissible paths not escaping a given neighborhood of the starting configuration contains a neighborhood of the starting configuration (Figure 2).

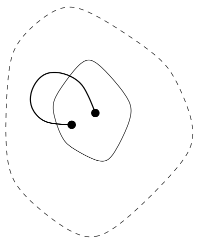


Figure 2: *Small-time controllable systems: the domain reachable without escaping a given neighborhood contains a neighborhood of the origin.*

Small-time controllability plays a central role in obstacle avoidance. Indeed, for small-time controllable systems, any collision-free path in CS may be approximated by a finite sequence of collision-free paths respecting the kinematic constraints. In other words, there exists an admissible collision-free path between two configurations if and only if both of them lye in the same connected component of the collision-free configuration space[2] CS_{free}. This means that the knowledge of the connected components of CS_{free} is sufficient to prove the existence of an admissible collision-free path between two given configurations. The route is open to devise deterministic and complete algorithms to solve the path planning problem.

The small-time controllability property does not hold for any system. A car moving only forward is locally controllable (the reachable set from a configuration contains a neighborhood of the configuration). It is not small-time controllable and deciding on the existence of an admissible collision-free path is still an open problem for this system.

Complexity Small-time controllability allows to forget the kinematic constraints to decide on the existence of an admissible path in CS_{free}. Nevertheless the kinematic constraints affect the combinatorial complexity of the admissible paths. Consider that CS_{free} is reduced to a long straight-line tube. Moving within the tube is easy for the robot arm: a single straight-line path is sufficient. Depending on the orientation of the tube in CS_{free}, the same task will be more difficult for the rolling bridge: the number of elementary pieces of the Manhattan paths may grow linearly when the width of the tube decreases. It will be even more difficult for a nonholonomic robot: the number of elementary pieces of admissible paths may grow exponentially with its degree of nonholonomy when the width of the tube decreases (e.g., the number of maneuvers required to park a car along a sidewalk varies quadratically with the inverse of the free space size).

3 Steering methods, visibility and roadmaps

Steering methods The decision part of the path planning problem (existence of a path) depends only on the controllability of the considered system. Now, to solve the problem completely (compute a path) we have to devise effective ways to steer the system. What we call a steering method is a procedure computing an admissible path between any two configurations in the absence of obstacles[3].

Any steering method respecting the kinematic constraints of the system is not necessarily a good one with respect to obstacle avoidance. For instance, a steering method for a car moving only forward applies also to a car moving both forward and backward. Nevertheless it is impossible to park a car by using only forward motions. A steering method induces a topology in CS that should account for the topology induced by the controllability property of the mechanical system. A steering method accounting for small-time controllability should verify the property illustrated in Figure 2. It is said to be stc. Devising stc steering methods is not necessarily a trivial problem (especially for nonholonomic systems). Moreover the choice is not unique. It

[2] Configurations touching an obstacle are not considered as collision-free. CS_{free} is an open domain in CS.

[3] In the path planning literature, authors usually refer to the notion of *local methods*.

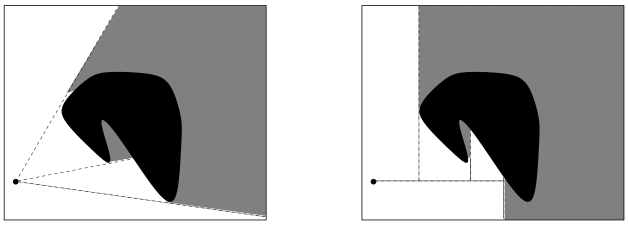
may affect the combinatorial complexity of the path planning algorithm.

This section aims to illustrate the importance of this choice through two examples of stc steering methods for a point moving on a plane. Both induce the same topology in CS. We will see that they differ by the combinatorics of the pavements induced on CS_{free}.

The first one (Steer_{lin}) consists in computing a straight-line segment between two points. The second one is Steer_{man}. Two points being given, Steer_{man} computes first an horizontal path from the left point until the right point is reachable by a vertical path. Both steering methods are symmetric and stc. Figure 3a shows the structure of the sets of points reachable with paths of fixed length.

A symmetric stc steering method Steer being given, a configuration is said to be *visible* from another one if the path computed between them by Steer lies in CS_{free}. The set of configurations visible from a given configuration constitutes its *visibility set*. The configuration is said to be the *guard* of its visibility set. Figure 3b shows visible sets of the same configuration, in the same environment, for both Steer_{lin} and Steer_{man}.

a: Reachable sets with paths of fixed length.



b: Visible sets in the presence of an obstacle (grey domains are not reachable).

Figure 3: *Visible sets of a configuration with* Steer_{lin} *(left) and* Steer_{man} *(right).*

Covering CS_{free} with visibility sets Let us consider a small-time controllable system together with a stc steering method. A set of guards is said to be *covering CS_{free}* if any configuration in CS_{free} is visible from at least one guard.

We address a first question: is there a finite set of guards covering CS_{free} ? The answer to this question is difficult and there exists today no general result. It depends on both the shape of CS_{free} and the considered steering method.

Let us illustrate the problem via four examples of 2-dimensional CS_{free} (see Figure 4). First consider Steer_{lin}. For the cases of Figures 4a and 4b, we may provide coverage of CS_{free}, e.g., by putting guards sufficiently close to convex vertices of the obstacles. The cases 4c and 4d contain an obstacle whose boundary piece is circular and tangent to a straight-line segment: there is no way to provide a finite number of guards covering CS_{free} for the case 4c, while this is possible for the case 4d (in this later example it is necessary to put at least one guard on the dashed line).

Consider the same environments with Steer_{man}. Finite coverage exists for the cases 4e, 4g and 4h while it does not exist for the case 4f.

Optimal coverage Now, we impose an additional constraint. We wish to provide coverage such that any guard does not see any other one. Withdrawing a single guard does not provide coverage anymore. For that reason we say that the set of guards provides *optimal coverage.*

Clearly, the set of guards providing optimal coverage is not unique. Taking a combinatorial point of view, a second question is: if there exists finite optimal coverage, are all the sets of guards achieving optimal coverage necessarily finite ? The answer is "no" in general.

The answer is "yes" if we consider Steer_{lin} and a polygonal CS_{free}: the maximal number of guards is bounded by the number of straight-line segments[4]. The answer is "no" if we consider a single circular obstacle (Fig. 5).

[4]Remark: computing the minimal number of guards constitutes the classical art gallery problem [23].

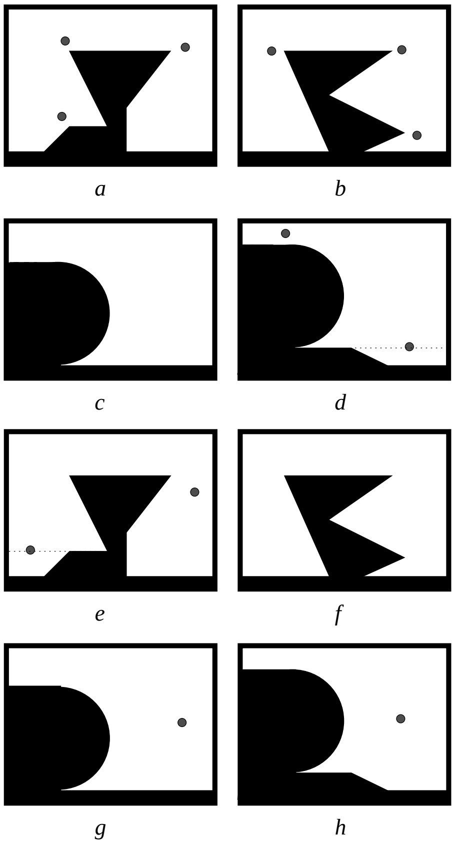


Figure 4: *Finite coverage using* Steer_{lin} *exists for the cases a, b and d. It does not for the case c. Finite coverage using* Steer_{man} *exists for the cases e, g and h. It does not for the case f.*

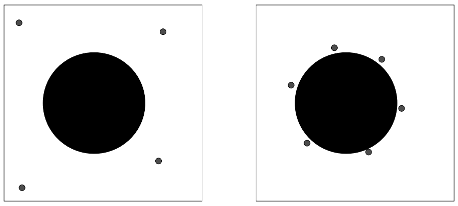


Figure 5: Steer_{lin}*: Examples of optimal coverage with 4 and 6 guards. The number of guards achieving optimal coverage is not a constant number. It may be unbounded.*

Also note that one cannot necessarily extract optimal coverage from any given set of guards providing coverage. More than that, finite coverage may exist while finite optimal coverage does not. In the example of Figure 6 finite coverage (with Steer_{lin}) necessarily requires two guards on the dashed lines. It is impossible to make one of them not visible from the other one. Nevertheless this example may be considered as a degenerated case since finite coverage requires guards on a domain whose dimension is strictly lower than the dimension of CS_{free} (the probability to select such guards randomly is null).

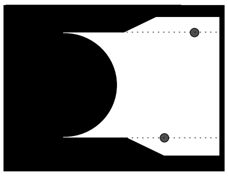


Figure 6: Steer_{lin}*: Finite coverage may exist while finite optimal coverage does not.*

Capturing CS_{free} connectivity from coverage In this section we show that coverage induces connectivity. Let Steer be a symmetric stc steering method in CS_{free}. Consider a given (finite or not) set of guards achieving coverage of CS_{free} together with a (finite length) path. The path is covered. Visibility sets are open sets. The path being a compact set, there is a *finite* number of guards whose visibility sets cover that path[5].

This property means that it is always possible to build a (finite or not) undirected graph accounting for the connectivity of CS_{free}. Two guards are connected if they are visible by Steer. At this level there does not exist necessarily a one to one mapping between the connected components of the graph and the connected components of CS_{free}. Nevertheless, it is always possible to complete the graph. When two guards belong to the same connected component of CS_{free} but not to the same component of the graph, and when their visibility sets intersect, we chose a configuration in the intersection set and we add it as a new node to the graph. Such a new configuration does not increase coverage,

[5]This gives rise to a notion of complexity of a path with respect to a given set of guards: the complexity of a path is the minimal number of guards necessary to cover the path.

it just makes the number of connected components of the graph decreasing. It is called a *connector*.

By this way, it is always possible to build a graph capturing the connectivity of CS_{free}, i.e. such that there is a one-to-one mapping between the connected components of the graph and the connected components of CS_{free}.

Everything is now in place to define a retraction method solving the path planning problem. The retraction function applied to a configuration just consists in selecting a guard seeing it. This is a well defined function since the set of guards covers CS_{free}. Finally some starting and goal configurations being given there exists an admissible collision-free path between them if and only if their retractions belong to a same connected component of the graph. The continuous dimension of the path planning problem is then transformed into a computational one.

The graphs above have been introduced in path planning literature with a computational point of view and especially with a probabilistic one. Indeed computing such roadmaps with complete deterministic algorithms is a difficult problem: for given CS_{free} and Steer, we do not know a priori if there exists finite coverage, and even if there exists, we do not know how to compute one of them in a deterministic way. The probabilistic approaches relax the completeness property and tend to cover CS_{free} at the best. Experiences show that they behave well for practical problems. This is the key of the success of the so-called *probabilistic roadmaps*.

When considering optimal coverage, the roadmap resulting from the construction above is a bipartite graph: the connector nodes are adjacent only to guard nodes and, from the definition of optimal coverage, a guard cannot see any other guard. Such roadmaps have been recently introduced as *visibility roadmaps* in [26].

The following section overviews probabilistic algorithms to compute roadmaps.

4 Probabilistic visibility roadmaps

Computing probabilistic roadmaps Consider a small-time controllable mechanical system. Computing a probabilistic roadmap for the system requires a collision checker and a symmetric stc steering method Steer that accounts for the kinematic constraints of the system.

The basic version of a probabilistic roadmap algorithm generates configurations randomly. As soon as a collision-free configuration is found, it is added as a new node to the graph. Then the algorithm checks if the current configurations previously generated are visible by Steer from the new configuration. Each time one of them is visible, a new edge is added, otherwise the new node creates a new connected component. A less expensive version consists in restricting the connectivity test to connect the new configuration to only one node for each connected component of the current graph. In that case, the resulting graph is a tree.

The more time consuming step of the algorithm is the call of Steer. Indeed the power of the algorithm is to avoid the compution of the visibility domains. Checking whether a configuration is visible from another one is done by checking if the path computed by Steer between them is collision-free. The number of calls to Steer then appears as a critical parameter that measures the combinatorial complexity of the construction.

Computing probabilistic visibility roadmaps The general principle above should be adapted to compute visibility roadmaps. We have seen that visibility roadmaps are bipartite graphs with two classes of nodes: the guards and the connectors. When a new collision-free configuration is (randomly) found, three cases may arise:

1. either it is not visible from any existing guard: then it is added as a new guard to the graph (it creates a new connected component),
2. or it is visible by guards belonging to separate connected components of the current graph: then it is added as a new connector and the corresponding connected components are merged by adding the edges between the new connector and the guards seeing it,
3. otherwise it is visible only by guards belonging to a same connected component and then it is rejected.

From a practical point of view, we have proposed a variant of the algorithm [26] allowing connectors to increase coverage of CS_{free}: connectors are added to the list of the guards. By this way covering CS_{free} is faster.

Comparison Figure 7 shows two examples of roadmaps computed for the same environment (polygonal CS_{free} with Steer_{lin}) with the same list of 250 random configurations. The first one is a basic roadmap. The second one is a visibility roadmap. In both cases the trees cover CS_{free} and capture its connectivity. On this example, the visibility roadmap algorithm ran 6 times faster than the basic roadmap algorithm.

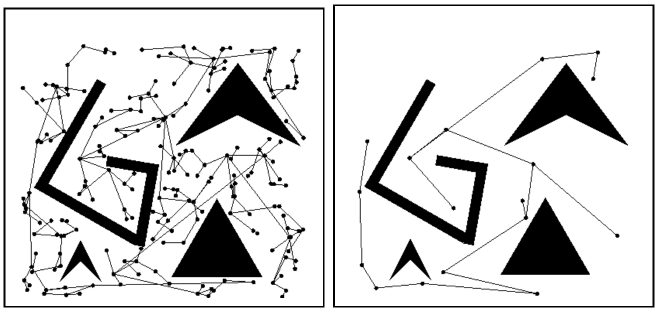


Figure 7: *A basic roadmap and a visibility roadmap.*

This simple example suggests a better performance of the visibility roadmap algorithm. They both cover and capture the connectivity of CS_{free}. The difference between the performances is due to the fact that visibility roadmaps keep a number of nodes smaller than basic roadmaps. The number of calls to Steer in the main loop of the algorithm is then smaller. This is illustrated on Figure 8.

However we do not succeed in proving formally a better behavior. Visibility roadmap algorithms may fail (in probability) in capturing the connectivity of CS_{free}. Guards may induce artificial narrow passages as shown in Figure 9. In that case, the probability to put a connector belonging to visibility sets of both guards is very low. With the same random sampling of CS_{free}, the visibility roadmap will certainly fail in capturing the connectivity of the space, while the basic roadmap will certainly succeed. Nevertheless, this

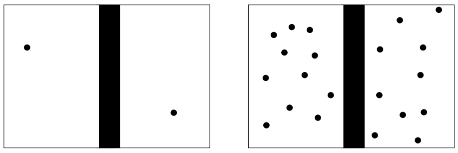


Figure 8: *Example of CS_{free} with two connected components. Any new collision-free configuration should be checked to know if it belongs to both connected components of CS_{free}. Checking this is much more expansive in the case of the basic roadmap (right) than in the case of the visibility roadmap (left).*

behavior may appear as a side-effect of the algorithm, since the probability to fall in this special case is clearly low (see [24] for details).

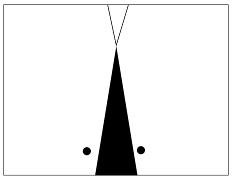


Figure 9: *A possible side-effect of the visibility roadmaps.*

Control of the algorithm A critical problem of the probabilistic algorithms is first to stop them, and second to get some information about the quality of the roadmap in terms of both coverage and connectivity. The structure of the visibility roadmap algorithm allows to control the quality of the roadmap in term of coverage.

We have seen that the algorithm consists in rejecting the configurations which are reachable only from nodes belonging to a same connected component of the roadmap. Therefore the expansion of the visibility set of a connected component of the visibility roadmaps may be a priori slower than those of the basic roadmaps. However, in addition to the computational gain discussed above, rejecting those configurations has an advantage: the number $\#try$ of failures in adding a new guard to the visibility roadmap allows to compute an estimation of the volume of the free-space remaining non covered. We model the percentage of the non covered free-space as the inverse of the number of failures before adding a new guard. The plain curve in Figure 10 shows the evolution of this estimated percentage of the covered free-space. The horizontal axis represents the number of guards, while the vertical one represents $(1 - \frac{1}{\#try})$. For the dotted curve the vertical axis represents the real percentages of the covered free-space (the curve is monotonic and increasing). The estimated percentage converges to the real one when the number of guards increases.

Guards achieve good coverage when the curve becomes horizontal. Such behavior may be detected within the algorithm by looking at the evolution of the number $\#try$. The algorithm stops when $\#try$ becomes constant (in probability). Of course we may also bound $\#try$ by a huge integer (a detailed version of the algorithm appears in [26]). In both cases, when the algorithm stops we get an estimation of the percentage of the non covered free-space volume.

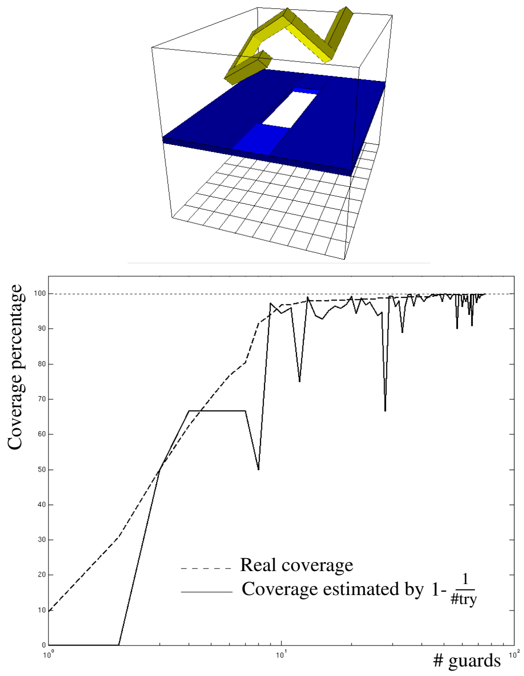


Figure 10: *Convergence of the visibility roadmap algorithm for a free flying body.*

5 Reading notes and comments

Probabilistic roadmaps have been introduced at the beginning of the 90's at both Stanford and Utrecht University [13, 25, 15]. PhD Thesis of L. Kavraki [12] and P. Švestka [27] state with details the seminal foundations of these new algorithms together with the first experimental results which are at the origin of their success. In this section we comment on several issues tending to understand their behavior on the basis of recent published papers.

Devising adequate steering methods In our presentation, we assumed that the considered steering methods are both symmetric and stc. The symmetry property allows to compute undirected roadmaps, while the stc property guarantees the convergence of the probabilistic algorithms.

Devising stc steering methods is not necessarily an easy task especially when we consider nonholonomic systems. An overview on this aspect may be found in [21].

In his PhD thesis P. Švestka relaxes the stc property. He shows that the steering methods may verify the following topological property: the set of configurations reachable by paths lying in an arbitrarily constrained domain contains a neighborhood in that domain (and not necessarily a neighborhood of the starting configuration as for the stc property). This property is important because it opens the range of applications to some locally controllable systems. Indeed the property holds for a car moving forward which is locally controllable but not small-time controllable. Due to the drift usually present in that kind of systems, the roadmap can no more be an undirected graph.

Algorithm analysis Capturing the connectivity of geometric spaces by sampling techniques is related to the percolation problem is statistical physics [11]. This problem asks deep and challenging questions in mathematics. Analyzing the probabilistic roadmap algorithms is a difficult problem. Several authors, mainly around Latombe's team, gave pertinent insights on this issue.

The notion of ϵ-goodness is introduced in [5]. It is related to the coverage property. CS_{free} is ϵ-good if the volume of the visibility set of any configuration in CS_{free} is greater than some fixed percentage $(1-\epsilon)$ of the total volume of CS_{free}. The authors prove that if CS_{free} is ϵ-good the probability that a (basic) roadmap does not cover CS_{free} decreases exponentially with the number of nodes. Moreover the number of nodes increases moderately when ϵ increases. Notice that the notion of ϵ-goodness is not strictly related to the existence of a finite set of guards achieving coverage. Indeed, for the cases illustrated in Figures 4a, 4b, 4g and 4h, CS_{free} is ϵ-good, while it is not for the cases 4c, 4d, 4e and 4f. Finite coverage may exist for non ϵ-good spaces. Moreover the number of guards achieving optimal coverage may be unbounded even for ϵ-good spaces (Fig. 5).

The notion of expansiveness introduced in [8] deals with connectivity. The proposed model is rather technical. It deals with the notion of narrow passages in CS_{free} and the difficulty to go through them. It is shown that for expansive CS_{free} the probability for a (basic) roadmap not to capture the connectivity of CS_{free} decreases exponentially with the number of nodes.

The clearance of a path is also a pertinent factor. In [14] a bound on the number of nodes required to capture the existence of a path of given clearance is provided. This bound depends also on the length of the path. In [28] the dependence on the length is replaced by the dependence on the number of visibility sets required to cover the path. Again the probability to fail in capturing the existence of a path decreases exponentially with the number of roadmap nodes.

Finally, the dependence on the dimension of CS is discussed in [10].

All these results are based on parameters characterizing the geometry of CS_{free}. The knowledge of these parameters would allow to control the algorithms. Unfortunately they are a priori unknown. As commented in [10], "*one could be tempted to use Monte Carlo technique to estimate the values of* [such parameters] *in a given free space, and hence obtain an estimate of the number of* [nodes] *needed to get a roadmap that adequately represents* [CS_{free}]. *But it seems that a reliable estimation would take at least as much time as building the roadmap itself*". An idea could be to try to estimate these parameters *during* the construction of the roadmaps. This is the underlying principle to control the visibility roadmap algorithm above. Generalizing it to estimate the ϵ-goodness, the expansiveness of the spaces or to detect narrow passages seems to be a promising issue to be investigated. The main challenge is to not degrade the performance of the algorithm with sophisticated computations.

On heuristics guiding probabilistic searches The same challenge appears to improve the basic roadmap algorithm behavior when facing difficult problems.

Capturing the connectivity of CS_{free} in the presence of narrow passages is one of them. In such contexts a first improvement has been introduced in [16]: the algorithm concentrates the search around small isolated components of the roadmaps. Another idea consists in sampling CS_{free} with configuration close to the obstacles. This can be done for free-flying systems by analyzing the contact configuration space, and by sampling contact subspaces defined by some geometric constraints like polyhedron vertex on polyhedron facet [3].

Contact space may be randomly sampled without using any explicit constraints. In [7] the method generates a pair of configurations separated by a random distance. When both configurations are either both collision-free or both in the collision space, they are rejected. Otherwise, the collision-free configuration is kept.

Another way to capture narrow passages is to exploit distance computations (an operation more expensive than a simple collision-checking): in a first step, configurations inducing small penetrations between bodies are accepted; then they are progressively moved to finally be collision-free [9].

Sampling close to the obstacles is a good strategy to capture the connectivity of narrow passages. Nevertheless it may induce undesirable side effects when the space is not very cluttered: in the example shown in Figure 5 sampling close to the obstacle requires more configurations than sampling far from the obstacle.

Experiences conduced on the visibility roadmaps show that the time gained by constructing small roadmaps may be used to concentrate the search on regions not connected to the roadmap. In other words, for a given fixed running time, the visibility roadmap algorithm will explore more space than the basic one. Visibility roadmaps behave rather well to capture narrow passages even if they do not have been devised to this end. The visibility roadmap algorithm runs 20 times faster than the basic roadmap algorithm to capture the connectivity of CS_{free} in the example of Figure 11 (see [26] for details).

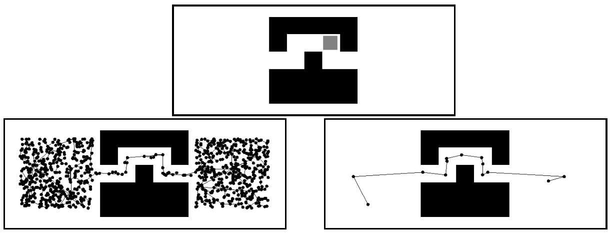


Figure 11: *Translating square (grey) with* Steer_{lin}*: the visibility roadmap algorithm (right) performs 20 times better than the basic roadmap algorithm (left) in capturing the connectivity of* CS_{free}.

On the choice of steering methods By definition, stc steering methods verify the topological property illustrated in Figure 2. This means that any path computed with a given method may be approximated by sub-paths computed with another one. Nevertheless we have seen that the combinatorics of a roadmap depends on the considered steering method. Depending on the shape of CS_{free} a steering method may perform better than another one. In Figure 12 we can see that, for the same example of environment, Steer_{lin} induces a smaller roadmap than Steer_{man}. Moreover the construction of the roadmap is much faster with Steer_{lin} than with Steer_{man}. However such a behavior is specific to the example. In the case of the rolling bridge in Figure 1, our experiments show a performance gain of 10 when using Steer_{man} instead of Steer_{lin}. This is a consequence of the special geometric structure of the workspace to be explored by a rolling bridge. Steer_{man} is more suitable than Steer_{lin} in that case.

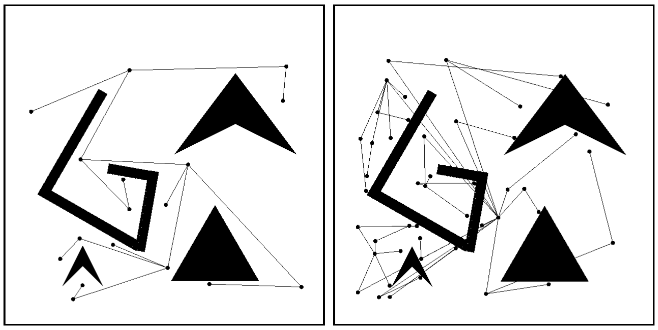


Figure 12: *Two visibility roadmaps computed with* Steer_{lin} *(left) and* Steer_{man} *(right).*

Therefore there is a priori no arguments to conclude on general results guiding the choice of a suitable steering method. This choice should be done after practical experiments on special classes of systems and environments. For instance, in [4] several steering methods working for free-flying rigid bodies are compared from a practical point of view. In particular a steering method consisting in decoupling translation and rotation is shown performing well.

6 Current directions

We began the integration of various probabilistic roadmap algorithms within a same software platform[6] Move3D. All the examples displayed in Figure 1 have been computed within Move3D by using the visibility roadmap algorithm. The generality constraint introduced as the main motivation of this work has been respected.

From a practical point of view, it remains to show that such techniques may face real size problems. The environment of Figure 13 represents a canonical example we are working on. The industrial installation is a stabilizer (subset of a plant in chemical industry). The geometric model was translated from PDMS geometric

[6] http://www.laas.fr/~nic/Move3D

data structures[7]. The crane was added within Move3D. Its workspace a priori covers all the environment. The model contains around 300.000 polygonal facets. Here the difficulty is to face the geometrical complexity of the environment in a reasonable time.

We are working on new hybrid collision-checkers combining the techniques developed in [22] and allowing to process polyhedra together with volumic primitives (e.g., spheres, tubes, torus…) as in [6].

On the other hand we are testing incremental approaches to roadmap constructions. The principle consists in partitioning the CS space of the crane into elementary regions. Then for each corresponding elementary workspace, we apply a filter on the geometric database to select the bodies to be handled by the collision-checker. Elementary roadmaps are computed in each CS regions and then merged together.

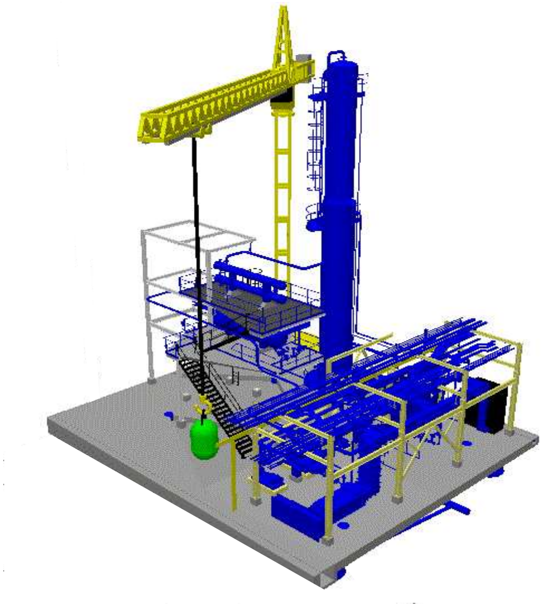


Figure 13: *A real-size problem.*

Another challenging issue is to provide the operator with facilities for combining several handling devices to carry freights. Path planning should be extended to handling task planning. This problem is often referenced as the manipulation planning problem. Its geometrical formulation [2] allows to apply probabilistic approaches. Results already appeared in [18] in the domain of digital actors animation and in [1] in the context of manufacturing.

[7]PDMS is a product of Cadcentre Ltd, partner of LAAS-CNRS, EDF and Utrecht University within the MOLOG project.

A priori such issues do not ask for new fundamental research. They require incremental research extending at best the existing state of the art.

From a theoretical point of view, the formal analyses referenced in Section 5 have opened the route toward a better understanding of the behavior and performance of probabilistic roadmap algorithms. Moreover additional work remains to be done to better control the probabilistic roadmap algorithms.

Acknowledgment We thank Carole Nissoux who has been the kingpin of Move3D.

References

[1] J.M. Ahuactzin, K.K. Gupta and E. Mazer. Manipulation planning for redundant robots: a practical approach. *Practical Motion Planning in Robotics*, K.K. Gupta and A.P. Del Pobil (Eds), J. Wiley, 1998.

[2] R. Alami, J.P. Laumond and T. Siméon. Two manipulation planning algorithms. *Algorithmic Foundations of Robotics (WAFR94)*, K. Goldberg et al (Eds), AK Peters, 1995.

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

[4] N. Amato, O Bayazit, L. Dale, C. Jones and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE International Conference on Robotics and Automation*, Leuven (Belgium), 1998.

[5] J. Barraquand, L. Kavraki, J.C. Latombe, T.Y. Li, R. Motvani and P. Raghavan. A random sampling scheme for path planning. *International Journal of Robotics Research*, Vol 16 (6), 1997.

[6] G. van den Bergen. Efficient Collision Detection of Complex Deformable Models using AABB Trees. *Journal of Graphics Tools*, Vol 2 (4), 1997.

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

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

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

[10] D. Hsu, L. Kavraki, J.C. Latombe and R. Motwani. Capturing the connectivity of high-dimensional geometric spaces by parallelizable random sampling techniques. *Advances in Randomized Parallel Computing*, P.M. Pardalos and S. Rajasekaran (Eds.), Combinatorial Optimization Series, Kluwer Academic Publishers, Boston, 1999.

[11] G. Grimmet. *Percolation.* Springer Verlag, 1989.

[12] L. Kavraki. Random networks in configuration spaces for fast path planning. PhD Thesis, Dept of Computer Science, Stanford University, 1995.

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

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

[15] L. Kavraki, P. Švestka, J.C. Latombe and M.H. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces, *IEEE Transactions on Robotics and Automation*, Vol 12 (4), 1996.

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

[17] L. Kavraki, M. Kolountzakis and J.C. Latombe. Analysis of Probabilistic Roadmaps for Path Planning, *IEEE Transactions on Robotics and Automation*, Vol 14 (1), 1998.

[18] Y. Koga, K. Kondo, J. Kuffner and J.C. Latombe. Planning motion with intentions. *ACM SIGGRAPH*, Orlando (USA), 1994.

[19] J.C. Latombe. Motion planning: a journey of robots, molecules, digital actors, and other artifacts. *The International Journal of Robotics Research*, Vol 18 (11), 1999.

[20] J.P. Laumond (Ed.). *Robot Motion Planning and Control.* LNCS 229, Springer Verlag, 1998 (out of print, available freely at `http://www.laas.fr/~jpl/`).

[21] J.P. Laumond, F. Lamiraux and S. Sekhavat. Guidelines in nonholonomic motion planning. In [20].

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

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

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

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

[26] T. Siméon, J.P. Laumond and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. (Submitted to Advanced Robotics Journal. A short version appeared in IEEE IROS, 1999).

[27] P. Švestka. Robot motion planning using probabilistic road maps. PhD Thesis, Dept of Computer Science, Utrecht University, 1997.

[28] P. Švestka and M. Overmars. Probabilistic path planning. In [20]