Path Deformation Roadmaps

Léonard Jaillet and Thierry Siméon

LAAS-CNRS, Toulouse, France
{ljaillet,nic}@laas.fr

Abstract. This paper describes a new approach to sampling-based motion planning with PRM methods. Our aim is to compute good quality roadmaps that encode the multiple connectedness of the Cspace inside small but yet representative graphs, that capture well the different varieties of free paths. The proposed approach relies on a notion of path deformability indicating whether or not a given path can be continuously deformed into another existing one. By considering a simpler form of deformation than the one allowed between homotopic paths, we propose a method that extends the Visibility-PRM technique [12] to constructing compact roadmaps that encode a richer and more suitable information than representative paths of the homotopy classes. The Path Deformation Roadmaps also contain additional useful cycles between paths in the same homotopy class that can be hardly deformed into each other. First experiments presented in the paper show that our technique enables small roadmaps to reliably and efficiently capture the multiple connectedness of the space in various problems.

1 Introduction

Robot motion planning has led to active research over the past decades [5] and sampling-based planning techniques have now emerged as a general and effective framework for solving challenging problems that remained out of reach of the previously existing complete algorithms. Today they make it possible to handle the complexity of many practical problems arising in such diverse fields as robotics, graphics animation, virtual prototyping and computational biology. In particular, the Probabilistic RoadMap planner (PRM) introduced in [4, 8] and further developed in many other works (see [2, 6] for a survey) has been conceived to solve multiple-query problems.

While most of the PRM variants focus on the fast computation of roadmaps reflecting the connectivity of the free configuration space, only few works [7, 9] address the problem of computing good quality roadmaps that encode the multiple connectedness of the space inside small graphs containing only useful cycles, ie. cycles representative of the varieties of free paths. Introducing such cycles is important for getting higher quality solutions when postprocessing queries, thus avoiding the computation of unnecessarily long paths, difficult to shorten by the smoothing techniques (e.g. [10, 13]).

S. Akella et al. (Eds.): Algorithmic Foundation of Robotics VII, STAR 47, pp. 19–34, 2008.
springerlink.com

Intuitively, the probability that a roadmap captures well the different paths varieties of $\mathcal{C}_{free}$ increases with its degree of redundancy. However, a direct approach attempting connections between all pair of nodes is far too costly and several heuristic-based connection strategies have been proposed to limit the number of redundant connections. A first way (e.g. [4]) is to limit the connection attempts of new samples to the k nearest nodes of the roadmap (or of each connected component). Another variant is to only consider nodes within a ball of radius r centered at the new sampled configuration (e.g. [1]). A more recent technique proposed in [7] only creates cycles between already connected nodes if they are k times more distant in the roadmap than in the configuration space. In all cases, the chances of capturing the different path varieties of $\mathcal{C}_{free}$ notably varies depending on the choice of the k or r parameter. Moreover it is difficult to choose with these heuristic sampling strategies the good parameter values for a given environment. This may result in a significant loss of performance regarding the roadmap construction process.

In this paper we present an alternative method to building compact roadmaps that are yet representative of the different varieties of free paths. The method only generates a limited number of useful cycles in the roadmap. Moreover it stops automatically when most of the relevant alternative paths have been found. Our approach relies on a notion of path deformability indicating whether or not a given path can be continuously deformed into another existing one. Compared with the standard notion of homotopy which is not directly suitable for our purpose because it relies on too complicated deformations (Sect. 2), we consider simpler and more easily computable deformations between paths (Sect. 3). This results in compact roadmaps capturing a richer set of paths than homotopy (Sect. 4). We describe in Section 5 a two-stage algorithm for constructing such (easy) path deformation roadmaps. The first stage uses Visibility-PRM [12] to construct a small tree covering the space and capturing its connected components as well as possible. The second stage aims at enriching the roadmap with new nodes involved in the creation of useful cycles. The key ingredient of this step is an efficient path visibility test used for the filtering of useless cycles that can be easily deformed into existing roadmap paths. Following the philosophy of Visibility-PRM, the second stage integrates a stop condition based on the difficulty of finding new useful cycles. Finally, our first experiments (Sect. 6) show that the technique enables small roadmaps to reliably capture the multiple connectedness of configuration spaces in various problems involving free flying or articulated robots.

2 Homotopy Versus Useful Roadmap Paths

First we informally discuss the relation between homotopy and the representative path varieties that it would be desirable to store in the roadmap. The capture of the homotopy classes of $\mathcal{C}_{free}$ corresponds to a stronger property

than connectivity. Two paths are called homotopic (with fixed end points) if one can be "continuously deformed" into the other (see section 3.1). Homotopy defines an equivalence relation on the set of all paths of $\mathcal{C}_{free}$. A roadmap capturing the homotopy classes means that every valid path (even cyclic paths) can be continuously deformed into a path of the roadmap. PRM methods usually do not ensure this property. Only the work of Schmitzberger [9] considers the problem formally and sketches a method for encoding the set of homotopy classes inside a probabilistic roadmap. However, the approach is only applied on two-dimensional problems and its extension is limited by the difficulty of characterizing homotopic deformations in higher dimensions.

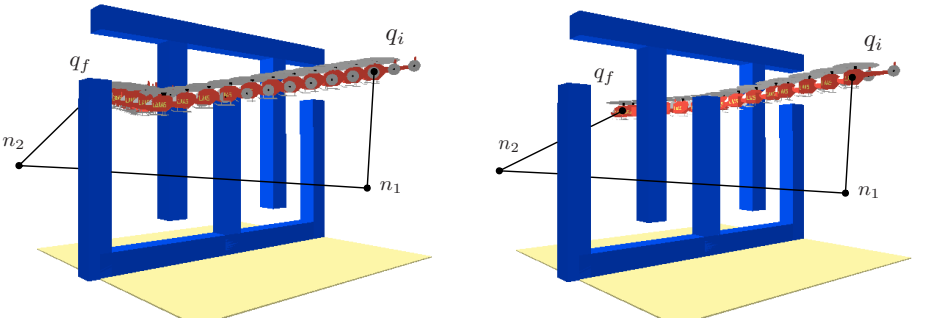

Fig. 1. Two examples of query for a 2 nodes graph (n_1-n_2). In the left picture, the solution path (q_i-n_1-n_2-q_f) extracted from the graph could be easily deformed into the displayed short path connecting query configurations (q_i, q_g) whereas a deformation in $\mathcal{C}_{free}$ would be much complex in the case of the right picture.

Moreover, as it was noted in [7] capturing the homotopy classes in higher dimensions may not be sufficient to encode the set of representative paths since homotopic paths (i.e. paths in the same homotopy class) may be too hard to deform into each other. This problem is illustrated by the example in Figure 1. Here $\mathcal{C}_{free}$ contains only one homotopy class. Therefore, an homotopy-based roadmap would have a tree structure, such as the simple 2 nodes (n_1,n_2) tree shown in the figure. While for the left query example, the solution path (q_i-n_1-n_2-q_f) found in the roadmap could be easily deformed into the displayed short path connecting query configurations (q_i, q_g), a free deformation would be much difficult to compute for the right example. Even if the topological nature of the two displayed paths is the same, their difference is such that it is preferable to store a representation of both paths in the roadmap. Generalizing this idea, we say that a roadmap is a good representation of the varieties of free paths if any path can be "easily" deformed into a path of the roadmap. This notion of simple path deformation is formalized below.

3 Complexity of a Path Deformation

In this section, after a brief reminder of the definition of a homotopic deformation, we propose a way to characterize classes of path deformations according to their complexity.

3.1 Homotopy

The homotopy between two paths is a standard notion from Topology (see [3] for a complete definition). Two paths τ and τ' in a topological space X are *homotopic* (with fixed end points) if there exists a continuous map $h : [0,1] \times [0,1] \rightarrow X$ with $h(s,0) = \tau(s)$ and $h(s,1) = \tau'(s)$ for all $s \in [0,1]$ and $h(0,t) = h(0,0)$ and $h(1,t) = h(1,0)$ for all $t \in [0,1]$.

Homotopy is a way to define any continuous deformation from one path to another. Next, we introduce a less general class of deformations, called *K-order deformations* characterizing specific subsets of homotopic deformations and that is used in section 4 for computing path deformation roadmaps.

3.2 K-Order Deformation

Definition 1. *A K-order deformation is a particular homotopic deformation such that each curve transforming a point of τ into a point of τ' is an angle line of K segments.*

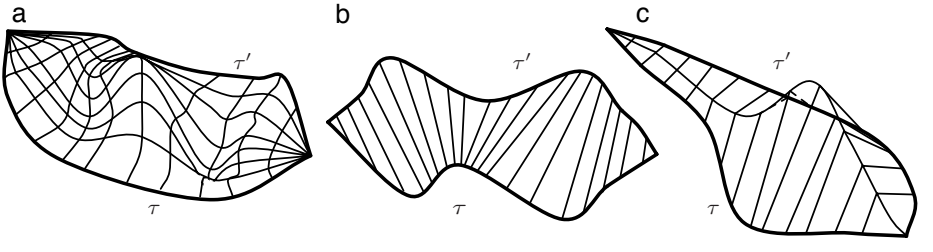

Fig. 2. (a) General homotopic deformation. (b) first order deformation: the deformation surface is a ruled surface. (c) Second order deformation: the deformation surface is obtained by concatenating two ruled surfaces.

Therefore, a first-order deformation surface describes a ruled surface and a K-order deformation is obtained by concatenation of K ruled surfaces. This is illustrated by Figure 2, which shows different types of path deformations: (a) is a general homotopic deformation whereas (b) and (c) respectively show 1st-order and a 2nd-order deformations.

Let D_i denote the set of i-order deformations. We clearly have $D^i \subset D^j$ for all $i < j$. Thus, the value K of the smallest K-order deformation existing between two paths is a good measure of the difficulty to deform one path into the other.

3.3 Visibility Diagram of Paths

It is important to note that a first-order deformation between two paths exists if and only if it is possible to simultaneously go through the two paths while maintaining a visibility constraint between the points of each path (see Figure 3). This formulation provides a computational way to test the existence of a first-order deformation, also called *visibility deformation* between two paths.

Let $\mathcal{L}_{lin}$ be the straight line segment between two configurations of $\mathcal{C}$. The parametric visibility function Vis of two paths (τ, τ') is defined as follows:

$$Vis : \begin{cases} [0,1] \times [0,1] \rightarrow \{0,1\} \\ Vis(t,t') \quad = 1 \text{ if } \mathcal{L}_{lin}(\tau(t), \tau'(t')) \in \mathcal{C}_{free} \\ Vis(t,t') \quad = 0 \text{ otherwise} \end{cases}$$

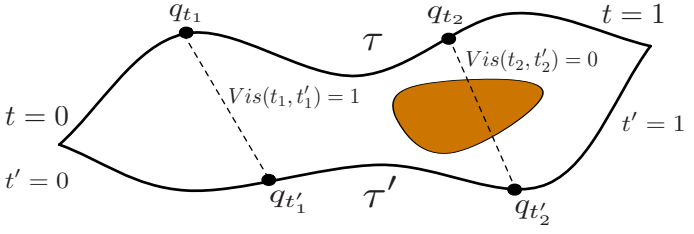

Fig. 3. The parametric visibility function of two paths evaluates the visibility between the points of each path

Then, the visibility diagram of paths (τ, τ') is defined as the two-dimensional diagram of the Vis function. It is illustrated by Figure 4 showing several examples of computed visibility diagrams with the corresponding paths.

Thanks to the visibility diagram, the visibility (i.e. first-order) deformation between two paths can now be expressed as follows: two paths (τ, τ') (with the same endpoints) are *visibility deformable* one into the other if and only if there is a path in their visibility diagram linking the points of parameters $(0, 0)$ and $(1, 1)$. Therefore it is possible to test the visibility deformation between two paths by computing their visibility diagram and then searching for a path in the diagram linking the points $(0, 0)$ and $(1, 1)$. In Figure 4, such a deformation is only possible for the last example (d).

4 K-Order Deformation Roadmap

In the previous section we have defined a way to characterize the complexity for two paths to be deformed one into the other. This formalism is now used to define, for a given roadmap, its ability to capture the different varieties of free paths of the configuration space.

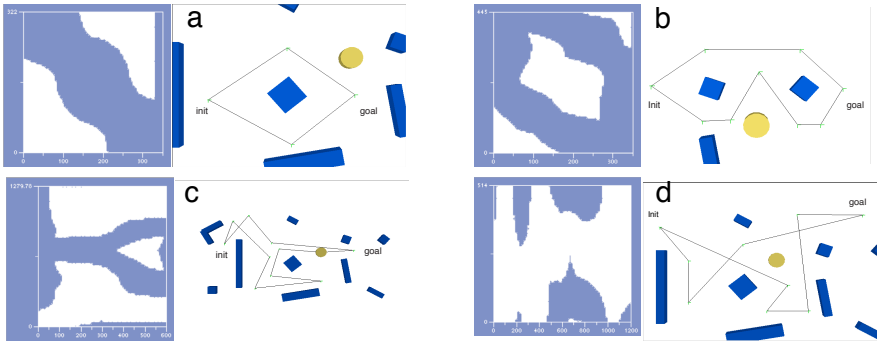

Fig. 4. Visibility diagrams for pairs of paths with the same endpoints. White areas represent regions where $Vis(t, t') = 1$. A visibility deformation is only possible in the last example (d), where a valid path linking the points (0,0) and (1,1) can be found in the visibility diagram.

Definition 2. *A roadmap R is a K-order deformation roadmap if and only if for any path τ of $\mathcal{C}_{free}$ it is possible to extract a path τ' from R (by connecting the two endpoint configurations of the paths) such that τ and τ' are K-deformable.*

This definition establishes a strong criterion specifying how the different varieties of free paths are captured inside the roadmap. One can also note that since a K-order deformation is a specific kind of homotopic deformation, any deformation roadmap captures the homotopy classes of $\mathcal{C}_{free}$. The following subsections present a computational method to construct such roadmaps.

4.1 Visibility Deformation Roadmap

We first define the notion of *Roadmap Connected from any Point of View* (called *RCPV* roadmaps) previously introduced in [9]. Then we establish that RCPV roadmaps are visibility (i.e. first-order) deformation roadmaps.

Visible Subroadmap

Let R be a roadmap with a set N of nodes and a set E of edges. If R covers $\mathcal{C}_{free}$, we can extract a set of nodes N_g (called guards) maintaining this coverage. Then, we can define for a free configuration q_v, the *Visible Subroadmap* $R_v = (N_v, E_v)$, as follows :

- N_v sublist of guards visible from q_v: $N_v = \{n \in N_g / \mathcal{L}_{lin}(q_v, n) \in \mathcal{C}_{free}\}$
- E_v, sublist of edges visible from q_v: $E_v = \{e \in E / \mathcal{L}_{lin}(q_v, e) \in \mathcal{C}_{free}\}$

Note that the notation $\mathcal{L}_{lin}(q_v, e) \in \mathcal{C}_{free}$ means that $\{\forall q \in e, \mathcal{L}(q_v, q) \in \mathcal{C}_{free}\}$. Examples of visible subroadmaps are presented in Figure 5.

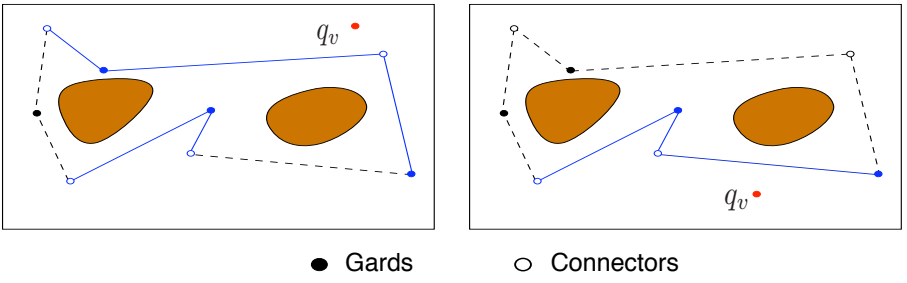

Fig. 5. Two examples of visible subroadmap from a given configuration q_v. On the left, the visible subroadmap is disconnected whereas it is connected on the right.

RCPV Roadmaps

Definition 3. *A Roadmap Connected from any Point of View (or RCPV roadmap) is such that for any configuration of $\mathcal{C}_{free}$, the visible subroadmap is connected.*

The following property establishes the link between RCPV roadmaps and visibility deformation roadmaps.

Property: A RCPV roadmap is a particular case of visibility deformation roadmap.

Sketch of proof: Let R be a RCPV roadmap and τ, a path of $\mathcal{C}_{free}$. τ can be partitioned into $2n-1$ successive paths:

$$\tau = \{\tau_{g_1} \oplus \tau_{g_1 \cap g_2} \oplus ... \oplus \tau_{g_i} \oplus \tau_{g_i \cap g_{i+1}} \oplus \tau_{g_{i+1}} \oplus ...\tau_{g_{n-1}} \oplus \tau_{g_{n-1} \cap g_n} \oplus \tau_{g_n}\}$$

with τ_{g_i} denoting the portion of path only visible from the g_i guard and $\tau_{g_i \cap g_{i+1}}$ the portion visible simultaneously from g_i and g_{i+1}. Since τ_{g_i} and g_i are by definition visible, it is possible to build a patch of ruled surface between them (Figure 6.a). Similarly, there is a patch of ruled surface between $\tau_{g_{i+1}}$ and g_{i+1}. Because R is a RCPV roadmap, any configuration $q_v \in \tau_{g_i} \cap \tau_{g_{i+1}}$ sees a path τ' connecting g_i to g_{i+1}. This property makes it possible to build a third patch of ruled surface between q_v and τ' (Figure 6.b). Finally, it is possible to fuse these three patches into a single ruled surface between $\tau_{g_i} \cap \tau_{g_{i+1}}$ and τ' (Figure 6.c). Thus, there exists a ruled surface (i.e. a visibility deformation surface) between the totality of τ and a path of the roadmap.

RCPV roadmaps are first-order deformation roadmaps. However, these roadmaps involve a high level of redundancy (see results section 6) and yet contain many useless cycles, especially in constrained situations. Therefore, to keep a compact structure we filter a part of the redundancy as explained in the following section. We will show that this filtering leads to a second-order deformation roadmap.

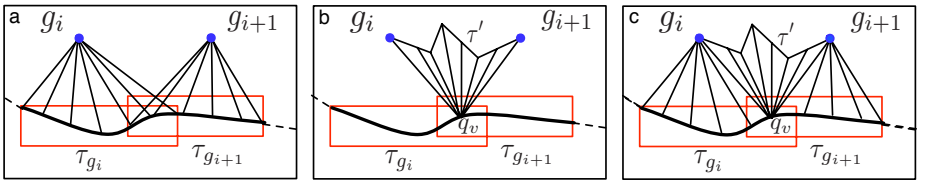

Fig. 6. A RCPV roadmap is a visibility deformation roadmap. (a) the visibility of the guards gives first patches of ruled surfaces. (b) the RCPV roadmap property guarantees the visibility of a roadmap path connecting two guards. (c) By construction, a global visibility deformation surface can be built.

4.2 Second-Order Deformation Roadmaps

Let R be a RCPV roadmap, $N_g \in R$ be a set of guard nodes ensuring the $\mathcal{C}_{free}$ coverage. Let us consider a pair of guards and τ, τ' two paths of the roadmap linking theses guards (i.e. creating a cycle) and visibility deformable one into the other. Then we have the following property:

Property: From a RCPV roadmap, the deletion of redundant paths τ' (i.e. visibility deformable into paths τ and connecting the same guards) leads to a second-order deformation roadmap.

Sketch of proof: Let us consider the partition of a free path τ, as defined in section 4.1. In that section we have shown that with a RCPV roadmap, one can extract a roadmap path τ' such that $\tau_{g_i} \cap \tau_{g_{i+1}}$ is visibility deformable into τ' (Figure 7.a). Now suppose that the redundant path τ' has been deleted as proposed above. It means that τ' was visibility deformable into another path τ'' which remains in the roadmap (Figure 7 b). Thus, by concatenation of the two ruled surfaces it is possible to build a second-order deformation surface between any path τ of $\mathcal{C}_{free}$ and a path of the roadmap (Figure 7 c).

Based on this notion of deformation roadmap, we describe below an algorithm for constructing such roadmaps.

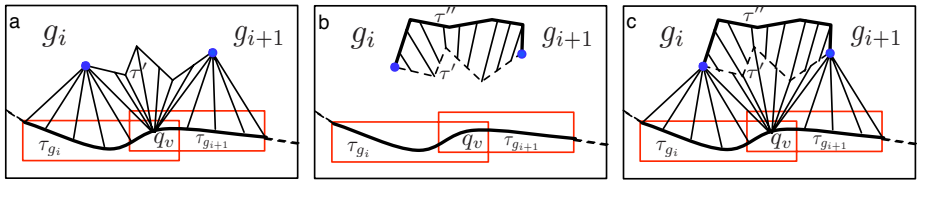

Fig. 7. Deleting redundant paths in a RCPV roadmap leads to a second-order deformation roadmap. (a) Visibility deformation for a RCPV roadmap. (b) A filtered path is itself deformable by visibility into a roadmap path. (c) By construction, there is a second-order deformation surface between a free path and a portion of roadmap.

5 Algorithm for Building Deformation Roadmaps

First, the roadmap is initialized with a tree structure computed with the Visibility-PRM method [12]. This ensures the coverage of the free space with a limited number of nodes and edges (i.e. no cycles). Then, instead of first building a RCPV roadmap and filtering in a second step the redundant cycles (as defined in section 4.2), the redundancy test is directly performed for efficiency purposes before each addition of a new cycle to the roadmap.

The pseudo-code of the algorithm used to build a second-order deformation roadmaps is shown in Figure 8. At each iteration a free configuration q_v is randomly sampled and the connectivity of the visible subroadmap is computed (*TestVisibSubRoadmap* function line 6). The evaluation of its connectivity is performed avoiding as much as possible the whole computation of the subroadmap. The redundancy test is only performed when the visible subroadmap is disconnected. For this test, we randomly choose two disconnected components of the subroadmap and pick among them the nearest guards n_1, n_2, from q_v. Then, we test whether there is a visibility deformation between the path $\tau = n_1 - q_v - n_2$ and a path of the roadmap (*TestRedundancy* function line 10). If such a visibility deformation exists, the configuration is useless with regards to the construction of a second-order deformation roadmap and is therefore rejected. The algorithm memorizes the number of successive failure since the last useful cycle inserted. This information is used to stop the iterations when the insertion of a new cycle becomes too difficult, meaning that most of the useful cycles are already captured by the roadmap.

```
PATH-DEFORMATION-PRM
input  : the robot A, the environment B, ntry_max, ntry_cycl_max
output : a Path Deformation Roadmap
1  G ← Visibility-PRM(A, B, ntry_max)
2  ntry ← 0
3  While ntry < ntry_cycl_max
4      q_v ← RandomFreeConfig(A, B)
5      ntry ← ntry + 1
6      If TestVisibSubRoadmap(G, q_v) = Disconnected
7          n_1 ← NearestGuard(q_v, Comp_1(G_v))
8          n_2 ← NearestGuard(q_v, Comp_2(G_v))
9          τ ← BuildPath(n_1, q_v, n_2)
10         If TestRedundancy (τ, n_1, n_2, G) = False
11             CreateCyclicPath(τ, G)
12             ntry ← 0
13         End If
14     End If
15 End While
```

Fig. 8. General algorithm for building a Path Deformation Roadmap

We next detail the algorithms used to establish the subroadmap connectivity (*TestVisibSubRoadmap* function) and to test the visibility deformation between pairs of paths (*TestRedundancy* function).

5.1 Visible Subroadmap

The method used to check the connectivity of a visible subroadmap from a given configuration q_v (*TestVisibSubRoadmap* function in the *Path-Deformation-PRM* algorithm) corresponds to the pseudo-code of Figure 9. First, the set of nodes visible from q_v is computed by testing whether the straight line segments linking q_v to each of the roadmap nodes are free. Then, we test in two phases the connectivity of these nodes from the point of view of q_v. First, we evaluate all the roadmap edges as potentially visible. Thus, two nodes are detected as disconnected if all the paths of the roadmap connecting them pass through at least one invisible node (*VisibleConnectivity* function line 8). If this fast test is not sufficient to establish the connectivity of the visible subroadmap, we establish it by computing the visibility of the edges linking the visible nodes (*VisibleConnectivity* function line 12). We describe in the next section how the visibility of an edge from a given configuration can be tested.

```
TestVisibSubRoadmap(G, q_v)
1  N_vis ← EmptyList
2  For all node n ∈ G
3       If VisibleNode(n, q_v)
4            AddToList(n, N_vis)
5       End If
6  Endfor
7  TestEdges ← False
8  If VisibleConnectivity(q_v, N_vis, G,TestEdges) = False
9       Return Disconnected
10 End If
11 TestEdges ← True
12 If VisibleConnectivity(q_v, N_vis, G ,TestEdges) = False
13      Return Disconnected
14 End If
15 Return Connected
```

Fig. 9. Algorithm testing the visible subroadmap connectivity from a given configuration q_v

5.2 Edge Visibility

Testing the visibility of an edge from a configuration q_v is equivalent to checking the validity of triangular configuration-space facets, defined by q_v and the two edge's endpoints (c.f. Figure 10). This visibility test possibly involves one or several facets depending on the topological nature of $\mathcal{C}$:

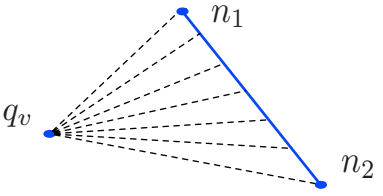

Fig. 10. Edge visibility: $n_1 - n_2$ is visible from q_v if the facet $\{q_v, n_1, n_2\}$ is valid

- If $\mathcal{C}$ is isomorphic to $[0,1]^n$ (the robot's degrees of freedom are only translations and/or bounded rotations) then the visibility test can be done by testing only a single facet in $\mathcal{C}$ (Figure 11.a).
- If $\mathcal{C}$ is isomorphic to $[0,1]^n \times SO(d)^m$ with $m > 0$ (one or more degrees of freedom are cyclic), the visibility test of an edge can lead to test several facets (Figure 11.b). In fact, a discontinuity occurs each time the distance between q_v and a configuration on the edge is equal to π according to a given degree of freedom.

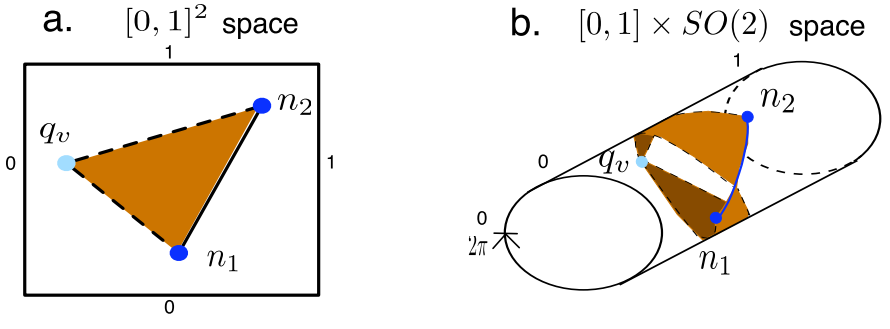

Fig. 11. Testing the visibility of an edge can lead to test one (a) or several (b) facets, depending on the topological nature of the configuration space

5.3 Elementary Facet Test

To test the validity of a facet we try to cover it entirely with free balls of $\mathcal{C}$ (Figure 12). First, the radii of the balls centered on each vertex of the facet are computed using a conservative method based on the robot kinematics and the distance of its bodies to the obstacles. If the balls are sufficient for covering the facet, then the algorithm returns that the facet is valid. Otherwise it is split into two sub-facets computed such that their common vertex is as far as possible from the regions already covered by the balls. The radius of the ball centered on this vertex is then computed. This dichotomic process is performed until the entire facet is covered or one vertex is tested as invalid.

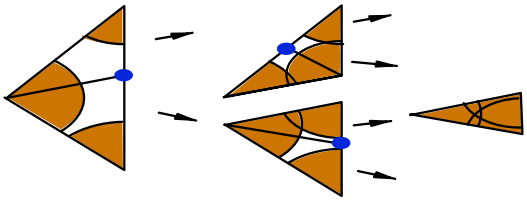

Fig. 12. Dichotomic covering of a valid facet with $\mathcal{C}_{free}$ balls

5.4 Redundancy Test

A disconnected subroadmap from the point of view of a configuration q_v can be reconnected by linking two of the subcomponents through q_v. Before performing such connection attempt, we test whether it may lead to a redundant path which could be filtered. To do so, first we build the path $\tau = n_1 - q_v - n_2$ with n_1, n_2 belonging to two distinct subcomponents. Then we test its visibility deformation into a roadmap path thanks to the *TestRedundancy* algorithm (line 10 in Figure 8). This algorithm is shown in Figure 13. Roadmap paths are iteratively extracted and tested according to their visibility deformation relatively to τ. This process starts with the shortest path found and stops when a visible deformation is possible (then the configuration is rejected) or when all the possible paths have been tested (then the configuration and the edges $n_1 - q_v$ and $n_2 - q_v$ are inserted).

```
TestRedundancy(τ, n1, n2, G)
1 τ' ← BestPath(n1, n2, G)
2 While τ' ≠ ∅
3     If VisibDeformation(τ, τ') = True
4         Return True
5     End If
6     τ' ← BestPath(n1, n2, G)
7 End While
8 Return False
```

Fig. 13. Visibility deformation test between a path τ and a roadmap path

The *VisibDeformation* function (line 3 of algorithm 13) tests whether two paths τ and τ' can be visibility deformed one into the other. This function is based on the grid based computation of the visibility diagram associated to the two paths. The deformation is only possible when there exists a path between the $(0, 0)$ and $(1, 1)$ points in this diagram (c.f. section 3.3). In practice, the whole diagram is not computed. The tests are limited to the grid cells visited during the A^* search of a valid path in the visibility diagram, incrementally developed during the search. This implicit search of the diagram notably limits the number of visibility tests to be performed (Figure 14) and highly speeds up the redundancy test.

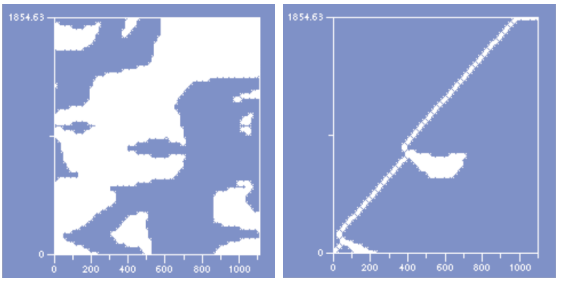

Fig. 14. Visibility diagram (left) and cells explored during the visibility deformation test (right)

6 Experimental Results

We implemented the algorithm for constructing (second-order) deformation roadmaps in the Move3D software platform [11]. The experiments reported below were performed on a 1.2GHz G4 PowerPC running on Mac OS-X. The performance results summarized in Table 2 correspond to average values computed over several runs of the algorithm.

The first experiment shown on Figure 15 compares the level of redundancy obtained in function of the algorithm used: (a), a minimum tree structure obtained with the Visibility-PRM, (b) a first-order roadmap (built without the filtering process) and (c) a second-order deformation roadmap (PDR) that captures the different varieties of paths while maintaining a compact structure.

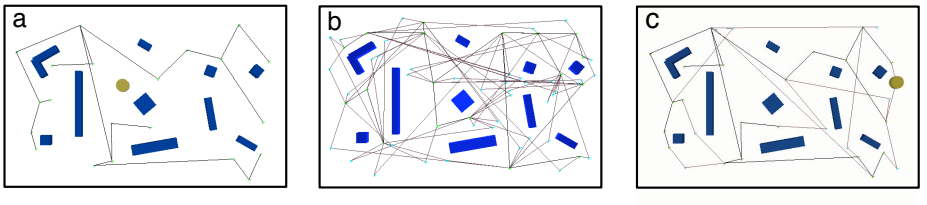

Fig. 15. Comparison between three algorithms of roadmap construction. (a) Visibility-PRM. (b), first-order and (c), second-order deformation roadmap.

The next set of experiments (Figure 16) presents the path deformation roadmaps obtained for a 2-dof robot evolving in complex environments. The first scene (a) requires 25 elementary cycles to capture the homotopy. Our method builds a roadmap capturing these cycles in only 109 seconds. The second scene (b) has a higher geometrical complexity (70 000 facets). The computing time (164 secs) reported in Table 2 shows that the algorithm can efficiently handle such geometrically complex scenes. One can also note that the resulting 2D roadmaps contain a very limited number of additional cycles compared to homotopy.

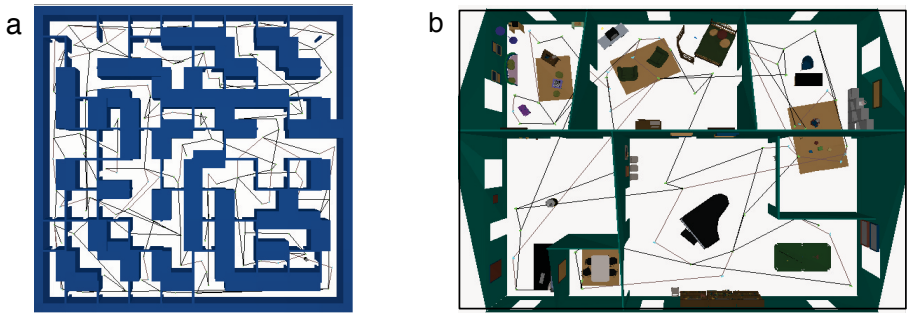

Fig. 16. Path Deformation Roadmaps for 2D environments: (a) a labyrinth with many homotopy classes. (b) an indoor environment with a complex geometry.

The third experiment (Figure 17) involves a narrow passage problem for a squared robot with 3-dof (two translations and one rotation). The robot has four ways to go through the narrow passage, depending on its orientation. Therefore the narrow passage corresponds to four homotopy classes in the configuration space.

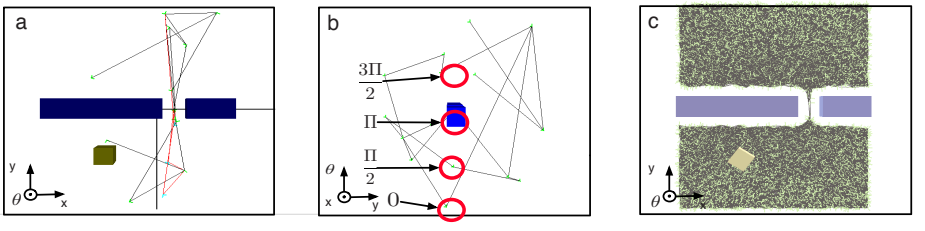

Fig. 17. Path Deformation Roadmap capturing the four homotopy classes for a rotating square and a narrow passage. (a) (x,y) view of the deformation roadmap, (b) (y,θ) view of the same roadmap showing the four kinds of passages found in $\mathcal{C}$, (c) comparison with the dense roadmap obtained with a classic k-nearest PRM.

Table 1. Homotopy classes found by a k-nearest PRM for the problem of Figure 17

		n_classes			time (s)		
	k	10	20	100	10	20	100
N	1000	0.1	0.2	1.2	6.4	9.3	33.2
	2000	0.1	0.6	1.6	33.2	43.5	110.0
	4000	0.8	1.0	2.8	246	336	455
	8000	1.4	2.4	**3.2**	2947	3295	**3819**

Table 1 presents results obtained with a traditional k-nearest PRM [4] for different couples (N, k) (with N, the number of roadmap nodes). The reported results (averaged over several runs) show that even for the densest and most redundant case ($N = 8000, k = 100$), the homotopy is not well captured (n_classes =

3.2/4) by the k-nearest PRM. Moreover, the large size of the computed roadmap results in a significant computing time (3819 secs) due to the amount of collision tests required for adding new nodes and edges. Comparatively, our method captures the four homotopy classes in only 37 secs. The high speed-up comes from the very compact size of the path deformation roadmap (only 12 nodes) which largely compensates the additional cost of filtering the useless redundant cycles.

The last set of experiments (Figure 18) involves 6-dof robots in 3D environments. In the first case (free flying robot), the free space has only one homotopy class. Thus, a roadmap based on homotopy would have a tree structure. The results show that our method makes it possible to build a compact roadmap (in 56 secs) while capturing a richer variety of paths than the homotopy. The second scene concerns a 6-dof manipulator arm where 6 additional nodes (and 12 edges) are added to the visibility roadmap (total time of 99 secs) to represent the complexity of the space. In both cases, the number of roadmap cycles, although limited, results into shorter paths during the query phase.

Finally, the performance results are summarized in Table 2 which also provides a break-up of the total computing time showing the respective contributions of the visibility tree building and the cycle addition stages.

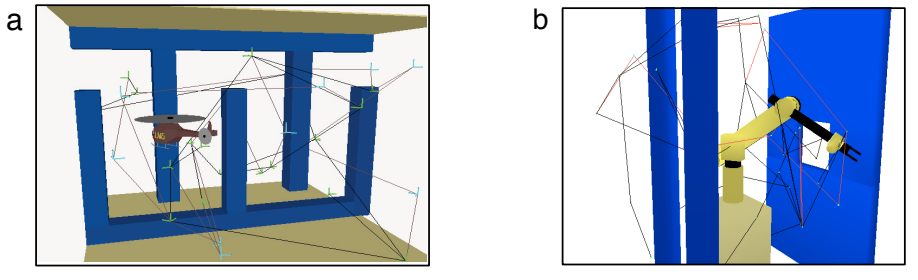

Fig. 18. Path Deformation Roadmaps for complex environments: (a) free flying robot, (b) 6-dof manipulator arm

Table 2. Computing time of the Deformation Roadmaps

						time repartition (%)			
	dof	nodes	edges	cycles	time (s)	Vis-PRM	SubRoadmap	Redundancy	Other
Laby	2	149	177	29	109	19	32	35	14
Indoor	2	66	83	18	164	25	20	49	6
Square	3	12	14	3	37	24	61	11	4
Helico	6	30	39	10	56	5	9	80	6
Arm	6	41	46	6	99	12	70	13	5

7 Conclusion

We have presented a general method to build compact PDR roadmaps with useful cycles representative of the different varieties of free paths of the configuration

space. The introduction of these cycles is important for obtaining higher quality solutions when postprocessing queries inside the roadmap. Our approach is based on the notion of path deformability indicating whether or not a given path can be easily deformed into another one. Our experiments show that the method enables small roadmaps to reliably capture the multiple connectedness of possibly complex configuration spaces. Several improvements remain for future work. First, the method has so far been tested for free flying and articulated robots with up to 6 dof. We need to further evaluate its performance for higher dof articulated robots. We would also like to further investigate the link between the varieties of free paths stored in the roadmap and the smoothing method used to shorten the solution paths when postprocessing queries. Finally, another improvement concerns the extension to robots with kinematically constrained motions requiring the use of a non-linear local method.

References

1. Bohlin, R., Kavraki, L.E.: Path planning using lazy prm. In: IEEE Int. Conf. on Robotics and Automation, pp. 521–528 (2000)
2. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of robot motion. MIT Press, Cambridge (2005)
3. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2002), `http://www.math.cornell.edu/~hatcher/AT/ATpage.html`
4. Kavraki, L.E., Svestka, P., Latombe, J.-C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)
5. Latombe, J.-C.: Robot Motion Planning. Kluwer Academic Publishers, Dordrecht (1991)
6. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2004-2005), `http://msl.cs.uiuc.edu/planning/`
7. Nieuwenhuisen, D., Overmars, M.H.: Useful cycles in probabilistic roadmap graphs. In: IEEE Int. Conf. on Robotics and Automation, pp. 446–452 (2004)
8. Overmars, M.H., Svestka, P.: A probabilistic learning approach to motion planning. In: Goldberg, K., et al. (eds.) Algorithmic Foundations of Robotics (WAFR 1994), pp. 19–37. A.K. Peters (1995)
9. Schmitzberger, E., Bouchet, J.L., Dufaut, M., Didier, W., Husson, R.: Capture of homotopy classes with probabilistic road map. In: IEEE/RSJ Int. Conf. on Robots and Systems (2002)
10. Sekhavat, S., Svestka, P., Laumond, J.-P., Overmars, M.H.: Multi-level path planning for nonholonomic robots using semi-holonomic subsystems. International Journal of Robotics Research 17(8), 840–857 (1998)
11. Siméon, T., Laumond, J.-P., Lamiraux, F.: Move3d: a generic platform for path planning. In: IEEE Int. Symp. on Assembly and Task Planning (2001)
12. Siméon, T., Laumond, J.-P., Nissoux, C.: Visibility-based probabilistic roadmaps for motion planning. Advanced Robotics Journal 14(6), 477–494 (2000)
13. Sánchez, G., Latombe, J.-C.: On delaying collision checking in prm planning - application to multi-robot coordination. International Journal of Robotics Research 21(1), 5–26 (2002)