Path Deformation Roadmaps: Compact Graphs with Useful Cycles for Motion Planning

Léonard Jaillet and Thierry Siméon
LAAS-CNRS, University of Toulouse
Toulouse, France

May 30, 2008

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 configuration space inside small but yet representative graphs that capture well the different varieties of free paths. The proposed Path Deformation Roadmaps (PDR) rely 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 (Siméon et al., 2000) to constructing compact roadmaps that encode a richer and more suitable information than representative paths of the homotopy classes. Path Deformation Roadmaps contain additional useful cycles between paths in the same homotopy class that can be hardly deformed into each other. Experimental results provided in the paper show that the technique enables small roadmaps to reliably capture the multiple connectedness of complex spaces in various problems involving free-flying and articulated robots in both 2D and 3D environments.

1 Introduction

Robot motion planning has led to active research over the past decades (Latombe, 1991). Sampling-based approaches 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.

The Probabilistic RoadMap planner (PRM) introduced in (Kavraki et al., 1996) and further developed in many other works (see Choset et al., 2005; LaValle, 2004 for a survey) has shown to perform well for a broad class of multiple-query problems, including in high-dimensional configuration spaces. The overall principle of PRM is to capture the connectivity of the collision-free space $\mathcal{C}_{free}$ by a set of one-dimensional curves stored in a precomputed roadmap. The roadmap is obtained by sampling robot configurations and subsequently connecting promising samples with valid local paths generated by a simple and fast local planner. Then, multiple path planning queries can be answered efficiently by simply connecting the query configurations and searching the augmented roadmap for a solution path, generally smoothed in a post-processing step to improve the quality of the solution. While PRM is successful for robots with many degrees of freedom and probabilistic complete, its performance degrades in the presence of narrow passages that require a prohibitively high density roadmap. A number of variants and extensions have been proposed to alleviate this problem and improve PRM performance, e.g. biasing sampling around obstacles (Amato et al., 1998; Boor et al., 1999; Hsu et al., 2003) or towards the medial axis (Wilmarth et al., 1999; Holleman and Kavraki, 2000; Lien et al., 2003), using free-space dilatation (Cheng et al., 2006; Saha and Latombe, 2005), visibility-based filtering (Siméon et al., 2000) or adaptive sampling (Kurniawati and Hsu, 2006; Rodriguez et al., 2006), exploiting search space information (Burns and Brock, 2005) or delaying collision checks (Bohlin and Kavraki, 2000; Sánchez and Latombe, 2002).

While most PRM variants focus on the fast computation of roadmaps reflecting the connectivity of the free configuration space, only few works (Schmitzberger et al., 2002; Nieuwenhuisen and Overmars, 2004; Geraerts and Overmars, 2006) address the problem of computing good quality roadmaps that encode inside small graphs the multiple connectedness of the space with a limited number of useful cycles, i.e. cycles representative of the varieties[1]of free paths. PRM often leads to dense roadmaps, whereas small graphs may be desirable in order to fasten both query time and roadmap updates. In contrast, Visibility-PRM (Siméon et al., 2000) produces very small roadmaps by rejecting most samples that lie in the visibility regions of existing guards. However, this pruning strategy leads to tree-like roadmaps that do not capture the multiple connectedness of the space. Introducing 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. Sekhavat et al., 1998; Sánchez and Latombe, 2002). Useful cycles also make the roadmap more robust to dynamic changes in the environment and may allow the planner to choose alternative routes for avoiding repetitive motions (Nieuwenhuisen and Overmars, 2004).

Intuitively, the probability that a roadmap captures well the different paths varieties of the free configuration space increases with its degree of redundancy. However, a direct approach attempting connections between all pairs of nodes is far too costly. Thus, several heuristic-based connection strategies are usually applied to limit the number of redundant cycles. A first way (e.g. Kavraki et al., 1996) is to restrict 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. Bohlin and Kavraki, 2000). A more recent technique proposed in (Nieuwenhuisen and Overmars, 2004) only creates cycles between already connected nodes if they are k times more distant in the roadmap than in the configuration space. This idea is also used in (Geraerts and Overmars, 2006) for creating high quality roadmaps for simple 2-3 dof robots in virtual environments. In all cases, the capture of the relevant path varieties notably varies depending on the choice of some parameter (e.g. k or r). Moreover it is difficult to choose with these heuristic sampling strategies the right parameter values for a given environment. This may result in a significant loss of performance of the roadmap construction process. A more formal technique (Schmitzberger et al., 2002) proposed for two-dimensional problems only considers cycles that encode the homotopy classes of the free space. Finally, other related works aim to increase the roadmap connectivity in constrained directions of the configuration space using a node connection strategy based on a Delaunay triangulation (Huang and Gupta, 2004) or exploit cycles for providing alternative routes in dynamic environments with mobile obstacles (van den Berg et al., 2005).

In this paper we present a new method to building compact roadmaps that are yet representative

[1]The term "path variety" is used in the paper to refer to a given class of similar paths.

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 excessively complex deformations (Sect. 2), we consider simpler and more easily computable deformations between paths (Sect. 3). This results in compact roadmaps that capture 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 builds on Visibility-PRM (Siméon et al., 2000) to construct a small tree covering the space and capturing its connected components as good 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 paths that can be easily deformed into existing roadmap paths. Following the philosophy of Visibility-PRM, the second stage also integrates a stop condition based on the difficulty of finding new useful cycles. Finally, some experimental results (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 endpoints) 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 (Schmitzberger et al., 2002) 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.

Moreover, as it was noted in (Nieuwenhuisen and Overmars, 2004) 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.

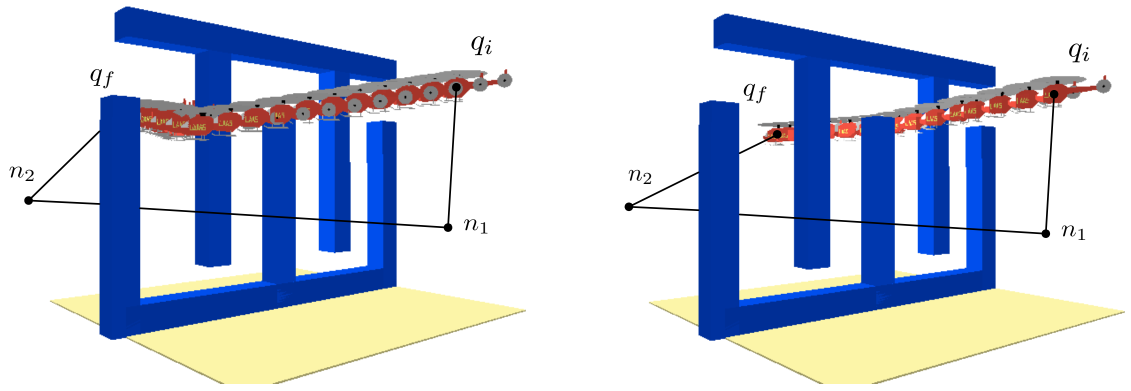

Figure 1: *Two examples of query for a 2 nodes graph (n_1-n_2). In the left picture, the solution path (q_i-n_1-n_2-q_f) extracted from the graph could be easily deformed into the displayed short path connecting query configurations (q_i, q_g) whereas a deformation in $\mathcal{C}_{free}$ would be much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

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 Hatcher, 2002 for a complete definition). Two paths τ and τ' in a topological space X are *homotopic* (with end points fixed) if there exists a continuous map $h : [0,1] \times [0,1] \to X$ 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 particular 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, ie. a piecewise linear curve, formed by K consecutive straight line segments.

Therefore, a first-order deformation surface describes a ruled surface [2] 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)

[2]A ruled surface is a surface that can be swept out by moving a straight line in space

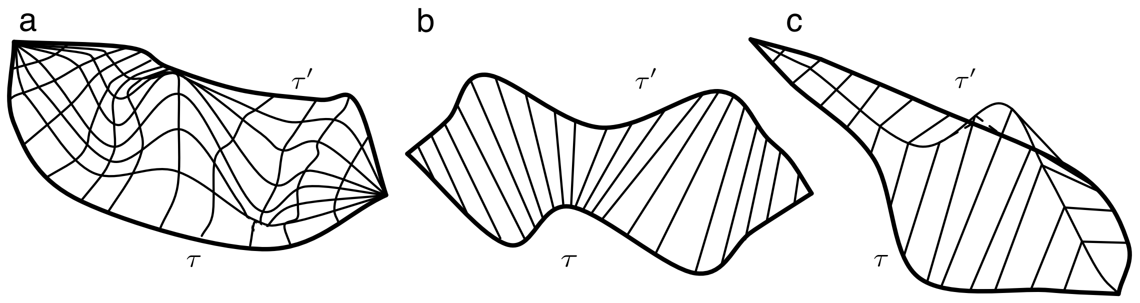

Figure 2: *(a) general homotopic deformation. (b) first order deformation: the deformation surface is a ruled surface. (c) Second order deformation: the deformation surface is obtained by concatenating two ruled surfaces.*

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:

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

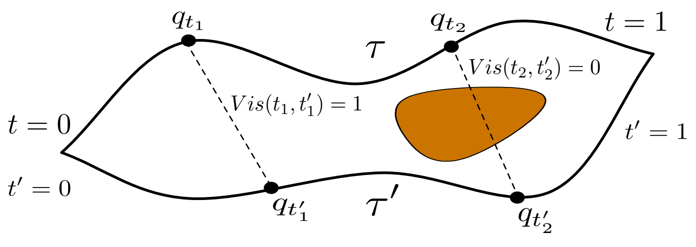

Figure 3: *The parametric visibility function of two paths evaluates the visibility between the points of each path.*

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

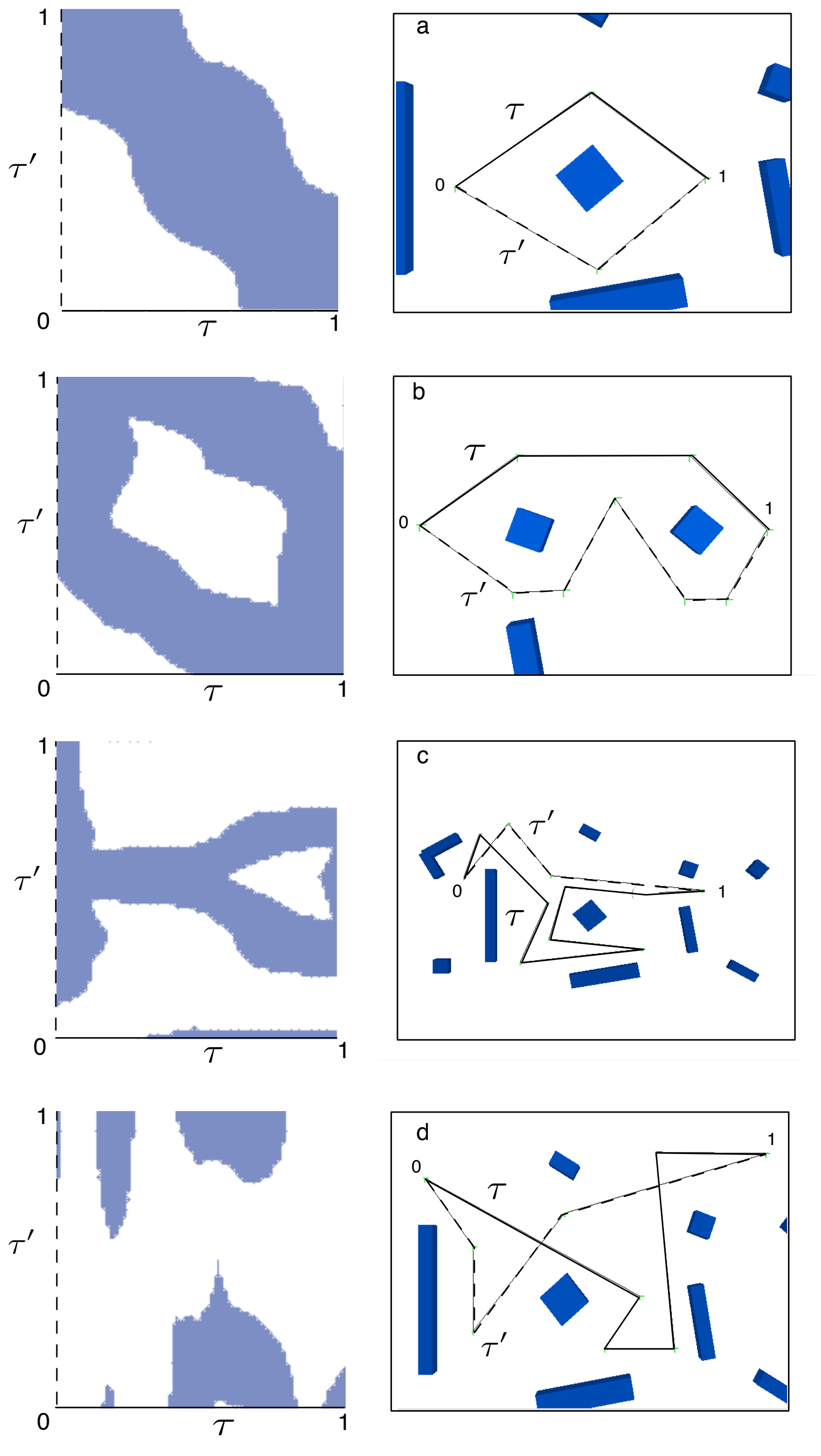

Figure 4: *Visibility diagrams for pairs of paths τ, τ' with the same endpoints. White areas represent regions where $Vis(t,t') = 1$. A visibility deformation is only possible in the last example (d), where a valid path linking the points (0,0) and (1,1) can be found in the visibility diagram.*

with the corresponding paths. Thanks to this 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 the two first examples (a-b) of Figure 4, there is no visibility deformation between the paths (τ, τ') since obstacles inside the cycle paths forbid any homotopic deformation. In the third example (c), an homotopic deformation between τ and τ' is possible, but the two paths are still not deformable by visibility. Finally a visibility deformation is only possible for the last example (d) where a valid path linking the points (0,0) and (1,1) can be found in the visibility diagram.

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 the ability of a given roadmap to capture the different varieties of free paths of the configuration space.

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'_R from R (by connecting the two extreme configurations of the path) such that τ and τ'_R 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 transformation, 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 (Schmitzberger et al., 2002). Then we establish that RCPV roadmaps are visibility (i.e. first-order) deformation roadmaps.

4.1.1 Visible Subroadmap

Let R be a roadmap with a set N of nodes and a set E of edges. Let also assume that R covers $\mathcal{C}_{free}$. The coverage property means that every configuration in $\mathcal{C}_{free}$ is visible from a node of R. Thus it is possible to extract from N a subset G of nodes (called guards) sufficient for 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: $N_v = \{g \in G / \mathcal{L}_{lin}(q_v, g) \in \mathcal{C}_{free}\}$
- 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.

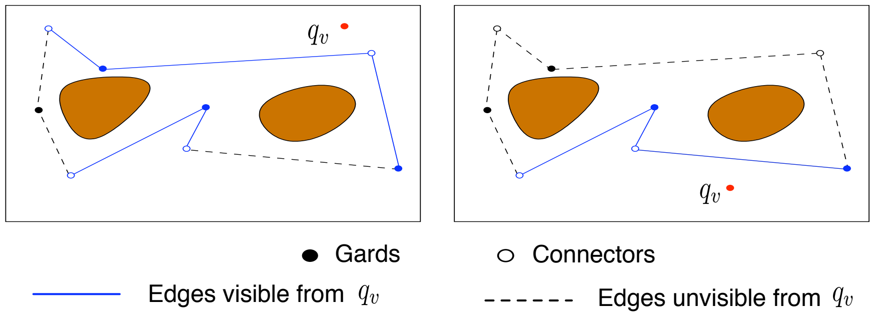

Figure 5: *Two examples of visible subroadmap from a given configuration q_v. On the left, the visible subroadmap is disconnected whereas it is connected on the right.*

4.1.2 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 $\mathcal{C}_{free}$. As a RCPV roadmap ensures the coverage of $\mathcal{C}_{free}$, τ can be covered by a given set of n guards, inducing its partitioning into successive elementary paths:

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

with τ_{g_i} denoting the portion of path visible from the g_i guard, $\tau_{g_i \cap g_{i+1}}$ the portion visible simultaneously from g_i and g_{i+1} (c.f. Figure 6). Note that τ can possibly go through the visibility region of a guard multiple times. Thus, we can have $g_i = g_j$ for $j \neq i-1$ and $j \neq i+1$ and τ is therefore partitioned into at least $2n-1$ portions.

Since τ_{g_i} and g_i are by definition visible, it is possible to build a patch of ruled surface between them (Figure 7.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'_R 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'_R (Figure 7.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'_R (Figure 7.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

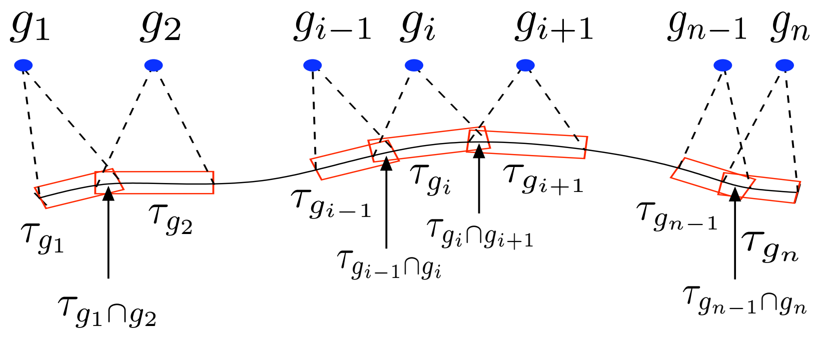

Figure 6: *Path decomposition in function of the portions visible from the guard nodes.*

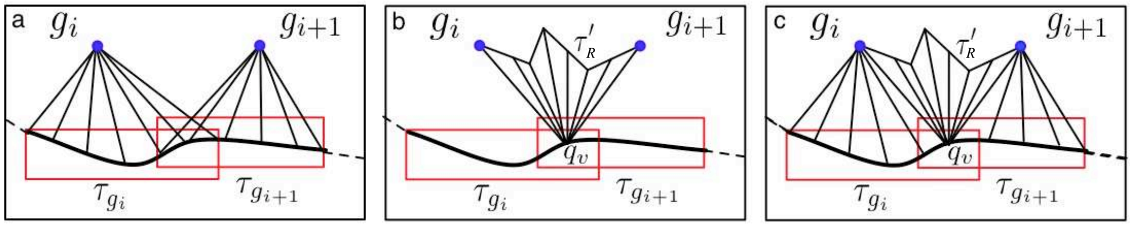

Figure 7: *A RCPV roadmap is a visibility deformation roadmap. (a) the visibility of the guard gives first patches of ruled surfaces. (b) the RCPV roadmap property guarantees the visibility of a roadmap path connecting two guards. (c) By construction, a global visibility deformation surface can be built.*

deformation roadmap.

4.2 Second-order Deformation Roadmaps

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

Property: From a RCPV roadmap R, the deletion of redundant paths τ'_R (i.e. visibility deformable into path 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 4.1.2. In that section we have shown that with a RCPV roadmap, one can extract a roadmap path τ'_R such that $\tau_{g_i} \cap \tau_{g_{i+1}}$ is visibility deformable into τ'_R (Figure 8.a). Now suppose that the redundant path τ'_R has been deleted as proposed above. It means that τ'_R was visibility deformable into another path τ''_R which remains in the roadmap (Figure 8 b). Thus, by concatenation of 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 8 c).

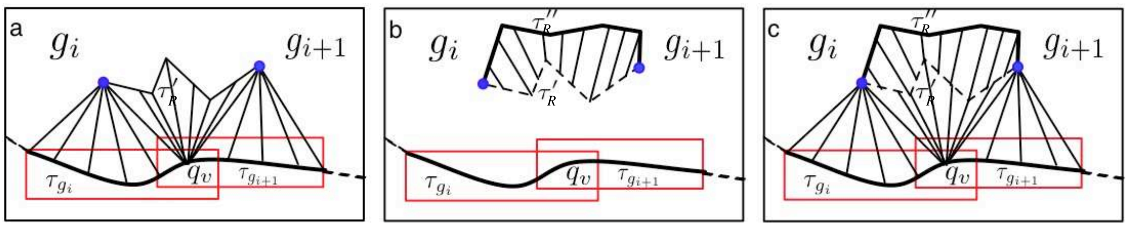

Figure 8: *Deleting redundant paths in a RCPV roadmap leads to a second-order deformation roadmap. (a) Visibility deformation between a path τ and a RCPV roadmap path τ'_R. (b) A filtered path τ'_R is visibility deformable into a roadmap path τ''_R. (c) By construction, there is a second-order deformation surface between a free path and a portion of roadmap.*

The above proofs are not constructive. The next section describes a sampling-based algorithm for constructing non-redundant graphs (referred to as "Path Deformation Roadmaps" in the rest of the paper) that tend to satisfy the second-order path deformation property.

5 Algorithm for building Path Deformation Roadmaps

The algorithm proposed for constructing Path Deformation Roadmaps (PDR) proceeds in two stages. First, it computes a small covering tree that captures the connectedness of the space. Then, during a second stage the initial tree is enhanced with useful cycles required for the multiple connectedness.

The initial covering tree is computed using Visibility-PRM. The pseudo-code of the algorithm is shown in Figure 9 (see Siméon et al., 2000 for a detailed description). Visibility roadmaps rely on a free-space structuring into visibility domains (i.e. sets of configurations connectable to a given guard by a valid local path). Computed guards are linked together via *connectors* located in their overlapping visibility regions. Such roadmaps can be constructed using a simple PRM variant : each free sample is added to the roadmap only if it cannot be connected to any existing node (i.e. guard) or if it connects at least two components (i.e. connector). The algorithm termination is controlled by the difficulty of adding a new guard ($ntry_{max}$ parameter) which relates to the quality of the roadmap in term of coverage (Siméon et al., 2000). The computed roadmap is a small tree (i.e. no cycles) capturing the free space coverage with a limited number of nodes and edges. However at this stage there is no guaranty concerning the deformability of $\mathcal{C}_{free}$ paths into roadmap paths.

Therefore, the visibility tree is enriched during the second stage with nodes and edges, creating the useful cycles required to obtain a path deformation roadmap. Instead of first building a RCPV roadmap and then filtering the redundant cycles (as defined in section 4.2), for efficiency purpose the algorithm directly performs the redundancy filtering before each addition of a new cycle to the roadmap.

The pseudo-code of the algorithm used to build a PDR is shown in Figure 10. At each iteration a free configuration q_v is randomly sampled and the connectivity of the visible subroadmap R_v seen by the sample is determined (*TestVisibSubRoadmap* function line 6). When the visible subroadmap is found to be singly connected, the sample can be directly rejected since any connections to the roadmap will obviously yield to useless cycles (see Figure 5 right). In the other

```
VISIBILITY-PRM

input     : the robot $A$, the environment $B$, $ntry_{max}$
output    : a roadmap $R$ with a tree structure
1  $ntry \leftarrow 0$
2  While $ntry < ntry_{max}$
3      $q \leftarrow$ RandomFreeConfig($A,B$)
4      $g_{vis} \leftarrow \emptyset$; $Connector \leftarrow False$
5      For all components $R_i$ of $R$
6          $g \leftarrow$ VisibleConfInComponent($q,R_i$)
7          If $g \neq \emptyset$
8              if $g_{vis} = \emptyset$
9                  $g_{vis} \leftarrow g$
10             Else
11                 NewConnector($q,g,g_{vis}$)
12                 $Connector \leftarrow True$
13             End If
14         End If
15     until $Connector = True$
16     If ($g_{vis} = \emptyset$)
17         NewGuard(q)
18         $ntry \leftarrow 0$
19     Else
20         $ntry \leftarrow ntry + 1$
21     End if
22 End While
```

Figure 9: *Visibility-PRM algorithm used to compute an initial tree in the PDR method.*

case, a reduncancy test of cycle paths possibly created by q_v must be performed. As explained in Sec. 5.1, the connectivity test of the visible subroadmap stops as soon as the subroadmap is found to be disconnected (thus avoiding as much as possible a whole connectivity test) and returns the two computed components $Comp_1$ and $Comp_2$. Then, the nearest nodes n_1, n_2 from q_v are selected inside these components and are used to test if there is a visibility deformation between the path $\tau = n_1 - q_v - n_2$ and a roadmap path linking n_1 to n_2 (*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. Otherwise, q_v is inserted in the roadmap as a new node and $n_1 - q_v$, $n_2 - q_v$ are also inserted as new edges. The algorithm memorizes the number of successive failures since the last useful cycle inserted. Similarly to the termination control of Visibility-PRM, this information is used to stop the iterations when the insertion of a new cycle becomes too difficult, i.e. when most of the useful cycles are already captured. From a convergence point of view, a roadmap computed with Path- Deformation-PRM tends toward a second-order deformation roadmap for sufficiently high values of its termination control parameter ($ntry\_cycl_{max}$).

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

```
PATH-DEFORMATION-PRM
input     : the robot $A$, the environment $B$, $ntry_{max}$, $ntry\_cycl_{max}$
output    : a Path Deformation Roadmap
1   $R \leftarrow$ Visibility-PRM($A, B, ntry_{max}$)
2   $ntry\_cycl \leftarrow 0$
3   While $ntry\_cycl < ntry\_cycl_{max}$
4       $q_v \leftarrow$ RandomFreeConfig($A, B$)
5       $ntry\_cycl \leftarrow ntry\_cycl + 1$
6       If TestVisibSubRoadmap($R, q_v$) $= Disconnected$
7           $n_1 \leftarrow$ NearestGuard($q_v, Comp_1(R_v)$)
8           $n_2 \leftarrow$ NearestGuard($q_v, Comp_2(R_v)$)
9           $\tau \leftarrow$ BuildPath($n_1, q_v, n_2$)
10          If TestRedundancy ($\tau, n_1, n_2, R$) $= False$
11              CreateCyclicPath($\tau, R$)
12              $ntry\_cycl \leftarrow 0$
13          End If
14      End If
15  End While
```

Figure 10: *General algorithm for building a Path Deformation Roadmap.*

5.1 Visible Subroadmap

The pseudo-code of the *TestVisibSubRoadmap* function (figure 11) outlines the lazy evaluation method used to check the connectivity of a visible subroadmap seen from a given configuration q_v. This two-stage process is also illustrated on Figure 12. Starting from the current roadmap (Fig. 12.a), all the edges are first initialized as potentially visible. The algorithm first checks the node visibility from q_v by testing the collision-freeness of straight line segments linking q_v to each roadmap nodes (Fig. 12.b). The set of non visible nodes is then used to speed up the connectivity test. Indeed, when a given node is labeled as non visible, all its edges can also be labeled as non visible from q_v. In most cases, this fast test is sufficient to establish the disconnectedness of the visible subroadmap without requiring more costly tests. Otherwise, the algorithm further proceeds by computing the visibility of edges linking the visible nodes (Fig. 12.c). Note that all edges are not systematically tested since the computation stops as soon as the visible subroadmap (Fig. 12.d) is found to be disconnected. Next section describes the visibility test between a whole edge and a given configuration.

5.1.1 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 13). The test can involve one or several facets depending on the topological nature of $\mathcal{C}$:

- 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 14.a).
- If $\mathcal{C}$ is isomorphic to $[0,1]^n \times SO(d)^m$ with $m > 0$ (i.e. one or more degrees of freedom are cyclic), the visibility test of an edge can lead to test several facets (Figure 14.b). A

```
TestVisibSubRoadmap($R, q_v$)

$N_{vis} \leftarrow$ EmptyList
**For** all node $n \in R$
    **If** VisibleNode($n, q_v$)
        AddToList($n, N_{vis}$)
    **End If**
**Endfor**
TestEdges $\leftarrow$ False
**If** VisibleConnectivity($q_v, N_{vis}, R$,TestEdges) $= False$
    Return Disconnected
**End If**
TestEdges $\leftarrow$ True
**If** VisibleConnectivity($q_v, N_{vis}, R$ ,TestEdges) $= False$
    Return Disconnected
**End If**
Return *Connected*
```

Figure 11: *Algorithm testing the visible subroadmap connectivity from a given configuration q_v.*

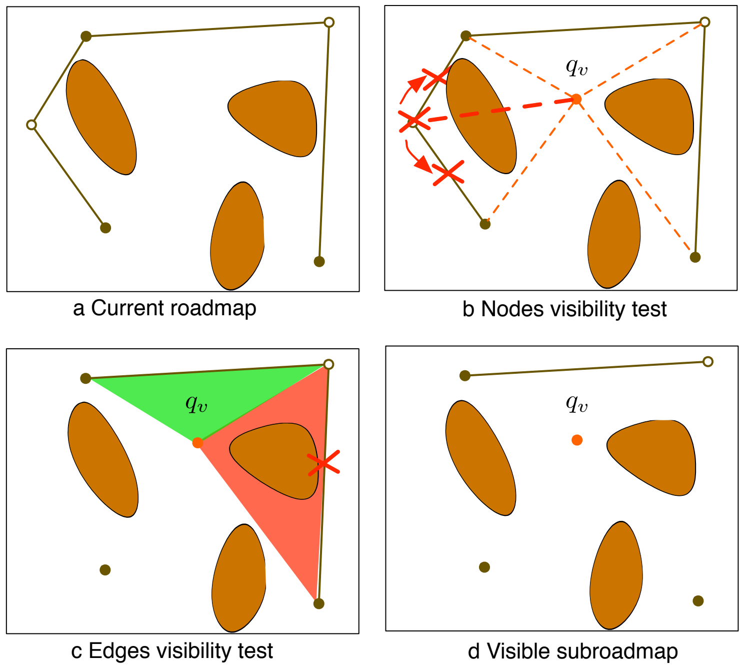

Figure 12: *Two-stage connectivity test of a visible subroadmap.*

discontinuity leading to a split in two facets 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.

5.1.2 Elementary Facet Test

To test the validity of a facet we try to cover it entirely with free balls of $\mathcal{C}$ (Figure 15). First, the radii of the free balls centered on each vertex of the facet are computed. If they are sufficient for

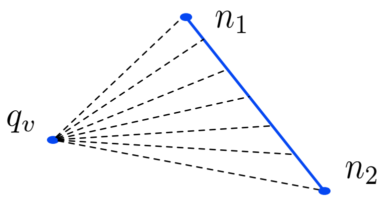

Figure 13: *Edge visibility: $n_1 - n_2$ is visible from q_v if the facet $\{q_v, n_1, n_2\}$ is valid.*

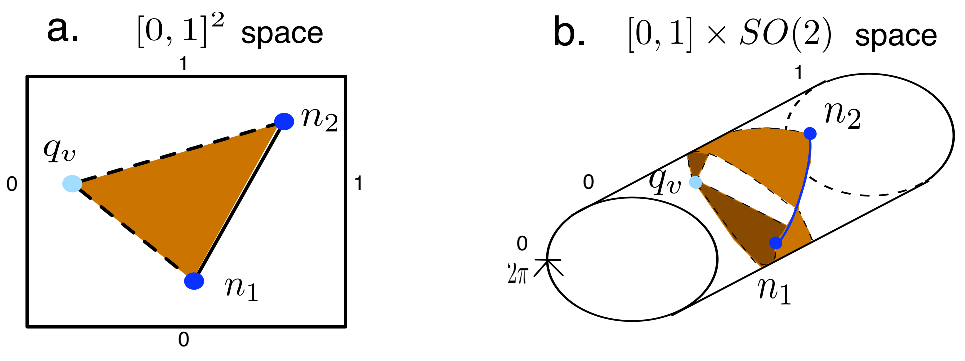

Figure 14: *Testing the visibility of an edge can lead to test one (a) or several (b) facets, depending on the topological nature of the configuration space.*

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 new vertex is then computed. This dichotomic process is performed until the entire facet is covered or one vertex is tested as invalid.

To compute the radius of a free ball centered on a vertex, we use a conservative method based on the robot kinematics and minimal distances of its bodies to the obstacles. The principle is similar to the one used for path collision detection with nonuniform step size (see LaValle, 2004 for a formal presentation and Jaillet, 2005 for the extension to the case of free balls of $\mathcal{C}$). In practice, such methods can however be too conservative when applied to complex robots with many rotational degrees of freedom. A discrete variant of the edge visibility test can be preferable to efficiently deal with such cases. It simply consists in discretizing the edge and checking the validity of the straight-line paths that connect q_v to the intermediate configurations along this edge. Another advantage of this discrete variant is to avoid the elementary facet decompositions phase (the switch of direction along the edge is automatically performed when computing the set of straight-line paths to be checked). Note that the discrete test was used in our experiments for the 6-dof manipulator example (see Fig. 21).

5.2 Redundancy Test

A disconnected subroadmap from the point of view of a configuration q_v can be reconnected by a path $\tau = n_1 - q_v - n_2$ with n_1, n_2 belonging to two distinct subcomponents. Such connection has to be performed only if it introduces cycles that are useful with regards to the construction of a

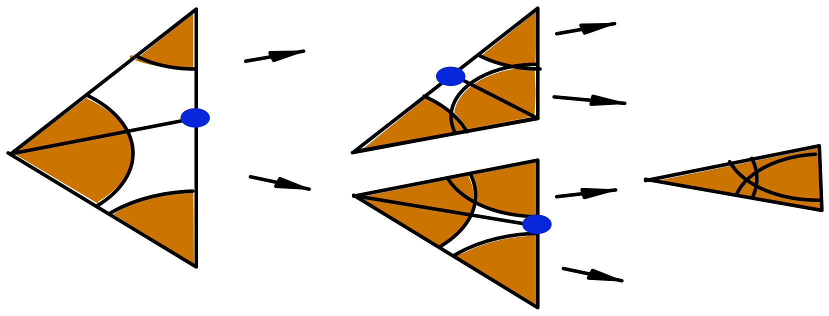

Figure 15: *Dichotomic covering of a valid facet with $\mathcal{C}_{free}$ balls.*

second order deformation roadmap. Testing the usefulness of adding a path τ is performed by the *TestRedundancy* algorithm (see pseudo-code in Figure 16). Roadmap paths linking nodes n_1 to n_2 are iteratively extracted and tested according to their visibility deformation relatively to τ. This process starts with the shortest path and stops when a visibility deformation is found (i.e. τ is useless and thus rejected) or when all the candidate paths have been tested (i.e. τ creates a useful cycle and it is inserted to the roadmap). In practice, only the k first shortest paths found in the roadmap (e.g. $k = 10$) are considered as candidate for the redundancy test. This filtering is justified by the fact that the longest paths have the less chances to be visibility deformable into the path τ. This is specially useful for complex environments where the roadmap may contain many cycles, resulting into possibly many paths between nodes n_1 and n_2. Finally, note that in the worst case, if the redundancy test fails to detect an existing deformation whereas it exists, then a "useless" cycle is added but the property of second order deformation roadmap stated in Definition 2 still holds.

```
TestRedundancy($\tau$,$n_1$,$n_2$,$R$)
 1  $\tau'_R \leftarrow$ BestPath($n_1$,$n_2$,$R$)
 2  While $\tau'_R \neq \emptyset$
 3      If VisibDeformation($\tau$, $\tau'_R$) $= True$
 4          Return $True$
 5      End If
 6      $\tau'_R \leftarrow$ BestPath($n_1$,$n_2$,$R$)
 7  End While
 8  Return False
```

Figure 16: *Visibility deformation test between a path τ and roadmap paths.*

The *VisibDeformation* function (line 3 of algorithm 16) tests whether two paths τ and τ'_R 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 noticeably limits the number of visibility tests to be performed (Figure 17) and significantly accelerates the redundancy test. Note that further speed up may be achieved using the lazy search technique proposed in (van den Berg and Overmars, 2007) combined with lifelong planning A* (Koenig and Furcy, 2004), aiming at further minimizing the number of grid cells tested for visibility.

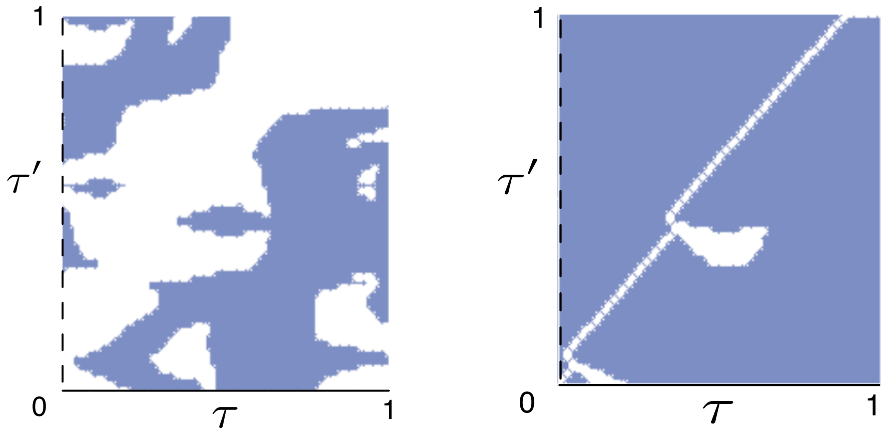

Figure 17: *Visibility diagram (left) and cells explored during the visibility deformation test (right).*

6 Experimental Results

We implemented the algorithm for constructing (second-order) deformation roadmaps in the Move3D software platform (Siméon et al., 2001). The experiments reported below were performed on a 1.2GHz G4 PowerPC running on Mac OS-X. The performance results summarized in Table 2 and 3 correspond to average values computed over several runs of the algorithm.

The first experiment shown on Figure 18 compares the level of redundancy obtained depending on 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 that captures the different varieties of paths while maintaining a compact structure. This clearly shows the interest of second-order deformation roadmaps (PDR) over first-order (RCPV) ones.

The next set of experiments (Figure 19) 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 makes it possible to build a roadmap capturing these cycles in only 109 seconds. The second scene (b) has a higher geometrical complexity (70 000 facets).

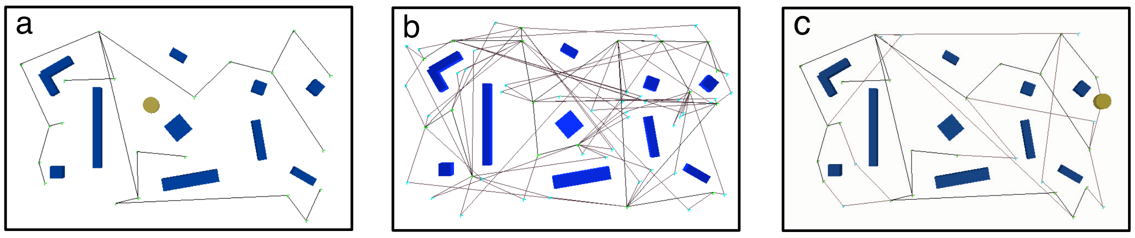

Figure 18: *Comparison between three algorithms of roadmap construction. (a) Visibility-PRM. (b), first-order and (c), second-order deformation roadmap.*

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 nodes compared to homotopy.

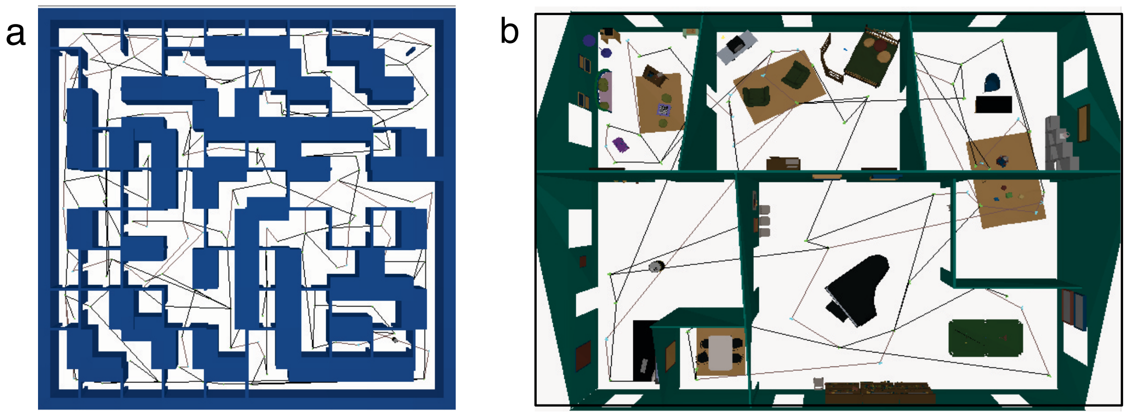

Figure 19: *Path Deformation Roadmaps for 2D environments: (a) a labyrinth with many homotopy classes. (b) an indoor environment with a complex geometry.*

The third experiment (Figure 20) 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.

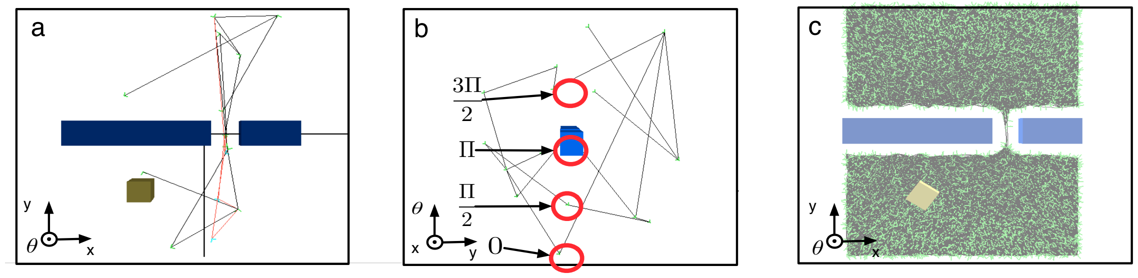

Figure 20: *Path Deformation Roadmap capturing the four homotopy classes for a rotating square and a narrow passage. (a) (x,y) view of the deformation roadmap, (b) (y,θ) view of the same roadmap showing the four kinds of passages found in $\mathcal{C}$, (c) comparison with the dense roadmap obtained with a classic k-nearest PRM.*

Table 1 presents results obtained with a traditional *k*-nearest PRM (Kavraki et al., 1996) for different couples (N,k) (with N, the number of roadmap nodes). The reported results (averaged over 10 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

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

		n_classes			time (s)		
	N	k =10	k =20	k =100	k =10	k =20	k =100
k-near PRM	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**
PDRoadmap	12	**4**			**37**		

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 21) 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.

Finally, Table 2 summarizes the performance results and Table 3 provides a break-up of the total computational effort by showing the respective contributions of the visibility tree building and the cycle addition stages.

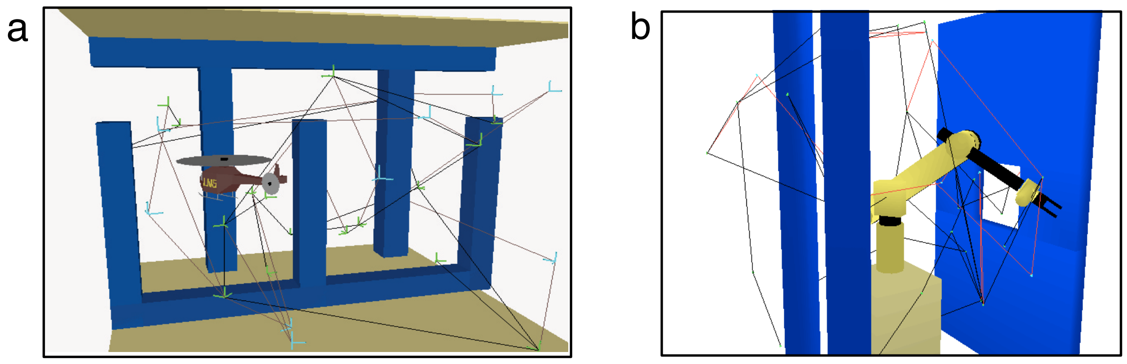

Figure 21: *Path Deformation Roadmaps for 3D environments: (a) free flying robot, (b) 6-dof manipulator arm.*

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

Table 2: General performance for the Roadmaps construction

Environments	2D Vis	2D 1-order	2D 2-order	Laby	Indoor	Square	Helico	Arm
Figure	18.a	18.b	18.c	19.a	19.b	20	21.a	21.b
dof	2	2	2	2	2	3	6	6
nodes	20	71	44	149	66	12	30	41
edges	19	121	34	177	83	14	39	46
cycles	0	51	14	29	18	3	10	6
time (s)	2	8	16	109	164	37	56	99

Table 3: Time repartition for the Roadmaps construction (in %)

Environments	2D Vis	2D 1-order	2D 2-order	Laby	Indoor	Square	Helico	Arm
Vis-PRM	100	19	13	19	25	24	5	12
SubRoadmap	-	75	15	32	20	61	9	70
Redundancy	-	-	66	35	49	11	80	13
Other	0	6	6	14	6	4	6	5

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 will 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 (e.g. nonholomic or closed chain robots) requiring the use of a non-linear local method.

Acknowledgment

This work was supported by the European projects MOVIE IST-20001-39250 and PHRIENDS IST-045359.

References

N.M. Amato, O.B. Bayazit, L.K. Dale, C. Jones, and D. Vallejo. Obprm: An obstacle-based prm for 3d workspaces. In P. Agarwal, L.E. Kavraki, and M. Mason, editors, *Robotics: The Algorithmic Perspective (WAFR1998)*, pages 155–168. A.K. Peters, 1998.

R. Bohlin and L.E. Kavraki. Path planning using lazy prm. *IEEE Int. Conf. on Robotics and Automation*, pages 521–528, 2000.

V. Boor, M.H. Overmars, and A.F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1018–1023, 1999.

B. Burns and O. Brock. Toward optimal configuration space sampling. *Proceedings of Robotics: Science and Systems*, 2005.

H.-L. Cheng, D. Hsu, J.-C. Latombe, and G. Sánchez-Ante. Multi-level free-space dilation for sampling narrow passages in PRM planning. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 1255–1260, 2006.

H. Choset, K.M. Lynch, S.Hutchinson, G. Kantor, W. Burgard, L.E.Kavraki, and S. Thrun. *Principles of robot motion*. MIT Press, 2005.

R. Geraerts and M.H. Overmars. Creating high-quality roadmaps for motion planning in virtual environments. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.

A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002. Available at `http://www.math.cornell.edu/~hatcher/AT/ATpage.html`.

C. Holleman and L.E. Kavraki. A framework for using the workspace medial axis in prm planners. *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1408–1413, 2000.

D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. *Proc. IEEE Int. Conf. on Robotics and Automation*, 2003.

Y. Huang and K. Gupta. A delaunay triangulation based node connection strategy for probabilistic roadmap planners. *Proc. IEEE International Conference on Robotics and Automation*, pages 908– 913, 2004.

L. Jaillet. *Méthodes Probabilistes Pour La Planification Réactive de Mouvements*. PhD thesis, Paul Sabatier University, 2005. Available at `http://robotics.cs.umass.edu/tc-apc/Main/Theses`.

L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

S. Koenig and D. Furcy. Lifelong planning A*. *Artificial Intelligence*, 155:93–146, 2004.

H. Kurniawati and D. Hsu. Workspace-based connectivity oracle: An adaptive sampling strategy for PRM planning. In S. Akella and et.al., editors, *Algorithmic Foundations of Robotics VII*. Springer–Verlag, 2006.

J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

S.M. LaValle. *Planning Algorithms*. Cambridge University Press, 2004-2005. Available at `http://msl.cs.uiuc.edu/planning/`.

J.-M. Lien, S.L. Thomas, and N.M. Amato. A general framework for sampling on the medial axis of the free space. *Proc. IEEE Int. Conf. on Robotics and Automation*, 2003.

D. Nieuwenhuisen and M.H. Overmars. Useful cycles in probabilistic roadmap graphs. *IEEE Int. Conf. on Robotics and Automation*, pages 446–452, 2004.

S. Rodriguez, S. Shawna, R. Pearce, and N.M. Amato. Resampl: A region-sensitive adaptive motion planner. *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2006.

M. Saha and J.C. Latombe. Finding narrow passages with probabilistic roadmaps: The small-step retraction method. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005.

E. Schmitzberger, J.L. Bouchet, M. Dufaut, W. Didier, and R. Husson. Capture of homotopy classes with probabilistic road map. *IEEE/RSJ Int. Conf. on Robots and Systems*, 2002.

S. Sekhavat, P. Svestka J.-P. Laumond, and M.H. Overmars. Multi-level path planning for nonholonomic robots using semi-holonomic subsystems. *International Journal of Robotics Research*, 17(8):840–857, 1998.

T. Siméon, J.-P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics Journal*, 14(6):477–494, 2000.

T. Siméon, J.-P. Laumond, and F. Lamiraux. Move3d: a generic platform for path planning. *IEEE Int. Symp. on Assembly and Task Planning*, 2001.

G. Sánchez and J.-C. Latombe. On delaying collision checking in prm planning - application to multi-robot coordination. *International Journal of Robotics Research*, 21(1):5–26, 2002.

J.P. van den Berg, D. Nieuwenhuisen, L. Jaillet, and M.H. Overmars. Creating robust toadmaps for motion planning in changing environments. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005.

J.P. van den Berg and M.H. Overmars. Kinodynamic motion planning on roadmaps in dynamic environments. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007.

S. Wilmarth, N.M. Amato, and P.Stiller. MAPRM: A Probabilistic roadmap planner with sampling on the medial axis of the free space. *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1024–1031, 1999.