# BBPRM: a behavior-based probabilistic roadmap method

Eva Simonin, Julien Diard

# BBPRM: A Behavior-Based Probabilistic Roadmap Method

Éva Simonin
Laboratoire d'Informatique de Grenoble
CNRS / INRIA Rhône-Alpes
Montbonnot Saint Martin, France

Julien Diard
Laboratoire de Psychologie et NeuroCognition
CNRS / Université Pierre Mendès France
Grenoble, France
Julien.Diard@upmf-grenoble.fr

***Abstract*—This paper focuses on the path planning problem. We offer an alternative to the probabilistic roadmap methods, from the perspective of modeling human or animal planning. In this context, hierarchies of representations are used to break down high-dimensional configuration spaces. We propose an approach for roadmap generation where low-level behaviors are used as articulations between level of the hierarchy. We also show how the obtained roadmap better represents low-level sensorimotor capabilities of the robot.**

## I. Introduction

Between robotics and biology, there is a bridge. Originally called cybernetics, this bridge is nowadays not symmetrical: while some robotics approaches get their inspiration from studies of living beings, relatively few works deal with transforming robotics algorithms into falsifiable models of animal behavior. The first direction is usually called biomimetic or bio-inspired robotics, while the second direction does not even have a precise name. Indeed, computational modeling in biology does not necessarily draw from robotics algorithms specficially, but more from general mathematical approaches.

Consider common robotics algorithms for mapping, like Simultaneous Localization and Mapping (SLAM), for perception, like probabilistic sensor fusion, for low-level control, like proportional-integral-derivative (PID) controllers, or for planning, like Probabilistic Roadmap Planning (PRM). Would they be plausible models of mapping, perception, low-level control and planning in animals?

Few of these have been treated yet. One of the exception is the current transfer of Maximum Likelihood Estimation (MLE) in sensor fusion to the study of human multimodal perception [1]. In this paper, we are interested in the study of the PRM method for planning, from the point of view of biology.

Here we consider the framework defined by Latombe, in which the main difficulty of planning is to represent the connectivity of the free space [2]. The roadmap methods describe how to build such a representation, in the form of a graph encoding a set of one-dimensional segments in the free space. Most of the current approaches build this graph in the configuration space instead of the worspace. Indeed, finding a path for a real robot in the workspace corresponds to finding a path for a point in the configuration space, thus making the searching of the path easier.

The Probabilistic Roadmap Method (PRM) is an efficient technique, as it can be applied to high dimensional configuration spaces. The particularity of this method is to imply a sampling of configurations at random [3], [4]. These configurations, called milestones, are linked by a local planner to build the roadmap. Then, in the obtained graph, paths are searched using classical graph-search algorithms.

This assumes that planning is tackled in a high-dimensional configuration space. In other words, planning is a part of a classical perceive-plan-act cycle: the first difficulty is to go from sensor activation to a high-dimensional internal space, the second is to compute paths in this complex internal space, the last one is to apply this path correctly, that is to say go back from this internal space to real actuators.

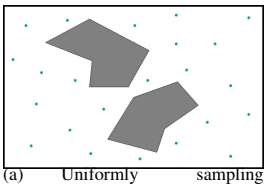
It is highly disputable that this can be a plausible foundation of models of animal behavior. Indeed, humans probably do not solve navigation tasks in this manner. Going from one chair to another in a room is surely not solved by considering the joint space of all articulation activation. Instead, the task is decomposed into separate phases, like standing up, walking, and sitting down. This suggests a hierarchical model, in which lower-limbs solve these sub-tasks, in small dimensional spaces, and provide solutions to higher-level processing in the form of behaviors. Given these, the 3-D space of the room can then be considered, and planning can be performed to provide a path to go from one chair to another, independently of the fine articulation activations needed at lower-levels to perform the basic behaviors. The high-dimensional space is thus tackled by being decomposed into sub-spaces, depending on the structure of the task. The question remains of how the articulation of the solutions in each sub-space can be formalized.

In this paper, we propose a modification of the PRM algorithm that takes into account this hierarchical assumption. We call our PRM variant BBPRM, for Behavior-Based Probabilistic Roadmap Method. It assumes low-level capabilities of the robot as *behaviors*, and we show how these can be incorporated into the PRM framework.

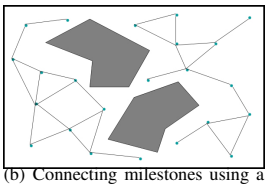
In BBPRM, during the node generation mechanism, we use a new type of local planner. Indeed, most of the milestones are generated as follows. Starting from a milestone $m_1$, a behavior $b$ is simulated for a given time $\Delta t$, and a new milestone $m_2$ is generated at the resulting position of the robot. In other words, the behavior $b$ acts as a local planner to set new milestones from existing ones and link them. The rest of the milestones are generated as in the basic PRM, i.e. by uniform random



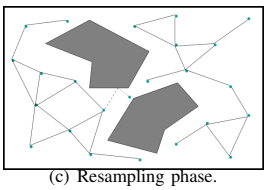
1-4244-2384-2/08/$20.00 



(a) Uniformly sampling milestones.



(b) Connecting milestones using a local planner.



(c) Resampling phase.

Fig. 1. The Probabilistic Roadmap Method principle.

sampling.

To illustrate the BBPRM concept, we have developed an experiment using a simulated robot. For simplicity, we have restricted ourselves to the case of a point-like robot in 2D. Its position in the workspace is encoded by its $x, y$ coordinates. Hence, the configuration space we consider is $\mathbb{R}^2$. We have defined and implemented a set of behaviors: "follow a wall", "follow a corridor" and "go straight". These behaviors require sensors to be implemented on the simulated robot: we simulated 8 simple proximity sensors located around the robot.

This experiment illustrates two main properties of BBPRM. Firstly, the built roadmap is a viable representation of space, in the sense that it is grounded in lower-level sensorimotor properties. Secondly, it outperforms the basic PRM for solving narrow passages, thanks to the underlying behaviors that are based on sensory information: because the sensors have limited range, obstacle boundaries are of special interest, contrary to free open space. This is translated in the final roadmap.

The rest of this paper is structured as follows. Section II summarizes the classical PRM method for path planning. Section III outlines the modifications we have made to the basic PRM algorithm in order to define BBPRM. Section IV and V detail respectively the implementation choices and results we obtained. Finally, Section VI offers a discussion of some theoretical aspects of BBPRM compared with the classical PRM, like computational cost and completeness of the planning.

## II. Probabilistic Roadmap Method

The Probabilistic Roadmap Method was developed by Kavraki et al. [3]. As for the deterministic roadmaps, the goal lies in capturing the connectivity of the free space with a set of segments. More precisely, it consists in building a graph whose edges correspond to segments in the collision-free configuration space.

### A. Roadmap generation

The basic method to build a probabilistic roadmap is composed of three phases. The first one consists in drawing milestones in the configuration space at random, according to a uniform distribution. Out of these sampled milestones, only the ones lying in the free space are kept and become the nodes of the graph. The sampling goes on until the number of milestones reach a value set by the programmer. At the end of this phase, a first set of nodes is obtained (see Fig. 1(a)).

The edges are generated during the second phase. This is done by searching, for a given milestone, if there are milestones close enough that can be linked with it by a path in the free space. A local planner is used to determine if a milestone can be linked to one of its neighbors. In most PRM implementations, this local planner checks if a straight line collision-free segment can link the two considered milestones. When the linkage is possible, the corresponding edge is added in the graph (see Fig. 1(b)).

The graph thus obtained does not necessarily succeed in representing the free space connectivity if this one includes narrow passages. Indeed, there is a risk is to obtain several non-connected sub-graphs if too few milestones have been sampled near or inside the narrow passages. The goal of the third phase, called the expansion phase, is to address this problem. During this phase, new milestones are created in the neighborhood of the narrow passages. The principle is to select some nodes in the graph which are probably close to narrow passages, according to some heuristic. From these selected milestones, the goal is to add new milestones that are close to them and that lie in the free space. Typically, these new milestones can be searched, starting from a given milestone, by following straight segments along a randomly chosen direction and bouncing on the encountered obstacles (see Fig. 1(c)).

There exist many variants in the algorithmic writings of the PRM concept. We summarize the roadmap generation method using a pseudo-code notation (see Alg. II.1).

**ALG. II.1** Algorithm of the probabilistic roadmap generation. The input is the configuration space geometric description, and the output is a graph defined by V, the set of nodes, and E, the set of edges of the graph. CLEAR is a subroutine that checks if a given configuration lies in the free space. LINK is a subroutine that checks if two given configurations can be linked using the local planner.

```
V ← ∅ and E ← ∅.
repeat
  q ← a uniformly sampled milestone
  if CLEAR(q) then
     Add q to V
     N_q ← a set of nodes in V that are
                                close to q
     for each q' ∈ N_q
        if LINK(q', q) then
           Add an edge (q,q') to E
```

### B. Roadmap use

Once it is constructed, the roadmap can be used for several planning requests. To answer a given request, the first step

consists in linking both the starting point and the goal point to milestones in the roadmap, using the local planner. Then, a path is searched in the graph between the two milestones found previously. The resulting path is a sequence of edges, each one corresponding to a segment in the free space found by the local planner during the roadmap construction. If needed, this global path can then be improved by smoothing methods.

### C. Finding narrow passages

The main difficulty in the PRM framework is to correctly represent the connectivity of environments that contain narrow passages. Those can be quantified using the expansiveness measure as defined by Hsu [5]: an environment that has a small expansiveness (small $\epsilon, \alpha, \beta$ measures) will possibly need a large number of milestones so as to have its connectivity correctly captured by the roadmap, thus reducing the main advantage of the PRM method.

The main solution to this "narrow passage" problem is to change the sampling method for generating milestones. For instance, methods have been designed that move the sampled milestones to the Voronoï of the free space [6]. Others have defined methods that try to sample milestones near the boundaries of obstacles:

- by changing the probability distribution used for sampling [7], [8],
- by allowing milestones to be generated slightly inside the obstacles and then be pushed back to the free space [9],
- or, more recently, by analyzing the workspace structure, according to the intuition that narrow passages in the configuration space may correspond to narrow passages in the workspace [10], [11], [12], [13].

## III. BBPRM PRINCIPLE

The trajectories given by PRM can be difficult to execute with a real robot. Indeed, these trajectories are usually made of straight segments in the configuration space, that come either from the local planner or from the expansion phase. These straight segments may become meaningless when translated into trajectories in the workspace. For instance, when considering a real robot, dynamic constraints may be violated at the junction of such configuration space straight segments. Approaches exists that deal with these difficulties, by smoothing the given configuration space trajectory, or by modifying it until it satisfies given constraints. However, these approaches do not consider the real physical capabilities of the robot as a central issue, but rather try to address this problem after a candidate path is given by the classic method.

We follow an alternative approach in BBPRM, by modeling the low-level sensorimotor capabilities of the robot as behaviors, and using these behaviors *during the roadmap generation*.

### A. Roadmap generation in BBPRM

*1) Generation of the milestones:* The milestones are generated by series of length $S$. The first milestone of each series is generated as in the basic PRM, i.e. by uniform random sampling over the configuration space. An initial robot direction is also uniformly sampled.

**ALG. III.1** Algorithm of the BBPRM roadmap generation. The input is the configuration space geometric description, and the output is a graph defined by V, the set of nodes, and E, the set of edges of the graph. CLEAR and LINK are the same subroutines as the ones used in PRM (see Alg. II.1). NOBLINK is a subroutine checking that no behavior link exists between two given milestones. SIMULATE_ROBOT is a subroutine that returns the milestone obtained by simulating the application of a behavior by the robot for a short time.

```
V ← ∅ and E ← ∅.
repeat
  q ← a uniformly sampled milestone
  if CLEAR(q) then
     Add q to V
     N_q ← a set of nodes in V that are
                                close to q
     for each q' ∈ N_q
        if LINK(q',q) and NOBLINK(q',q)
        then
           Add an edge (q, q') to E
     repeat
        q_b ← SIMULATE_ROBOT(q)
        Add q_b to V
        Add an behavior edge(q, q_b) to E
        N_qb ← a set of nodes in V that
                          are close to q_b
        for each q' ∈ N_qb
           if LINK(q',q_b) and NOBLINK(q',q_b)
           then
              Add an edge (q_b, q') to E
        q ← q_b
```

The rest of the milestones of a series is generated using behaviors. Starting from the milestone number $i-1$, the sensory situation is computed, a behavior is chosen accordingly, which computes a new robot direction and shifts the position by a small increment. The position and direction are updated $k$ times before generating the milestone number $i$.

*2) Linkage of the milestones:* We use two types of links between milestones. The milestones in one series are linked by "behavior links", which state that to go from one milestone to another, a given behavior was applied. The milestones between different series are linked in the same manner as in the basic PRM: for each milestone, the distances to milestones in other series are computed. When this distance is less than a threshold $\tau$, and the link does not cross any obstacle boundary, the link is added to the roadmap.

We summarize the roadmap generation in BBPRM using a pseudo-code notation (see Alg. II.1). The reader is invited to compare Alg. II.1 and Alg. III.1.

### B. Roadmap use in BBPRM

The roadmap is used in a similar manner is in PRM. First, the starting and goal points are linked to the roadmap. If there is a milestone in the neighborhood of these points (distance less

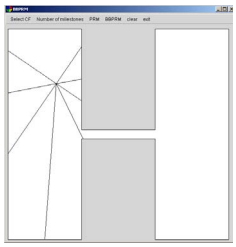


Fig. 2. Example of the simulated sensors.

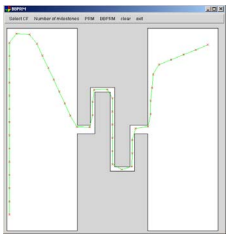


Fig. 3. Example of the generation and linkage of a series of milestones.

than $\tau$), they are linked to this milestone using the classic local planner. If there is no milestone within $\tau$ distance, behaviors are applied, from the start or goal point, until getting within $\tau$ of a milestone. The start and goal points being connected to the roadmap, a path is searched in the graph. The global path thus obtained is a sequence of paths that are either short straight segments or the application of a behavior.

## IV. Experiment

We now detail an experiment conducted for validating the BBPRM concept. We begin by describing the way the robot sensors and behaviors were simulated.

### A. Simulation of the sensors

The sensors we defined have been inspired by the sensors of the Khepera robot (K-Team, Switzerland). The robot has 8 such sensors, each facing a different direction. In this direction, they compute the distance to the nearest polygon edge that describe the obstacles. If the returned value is greater than some fixed sensor range, the sensor returns that there is no obstacle that can be seen by the robot. In the implementation, this sensor range was set to 4 units of distance (the configuration space is approximately in a $100 \times 100$ bounding box).

We have assumed a point-like robot in a two-dimensional configuration space. For the purpose of simulating the sensors, we also defined the direction of the robot, in order to compute the absolute angles of each sensor. The sensors are set around the robot using angles of $\{-135, -90, -45, -15, 15, 45, 90, 135\}$ degrees, relative to its direction. Negative angles refer to sensors on the left side of the robot, positive angles to sensors on its right.

Fig. 2 shows an example of the simulation of the sensors. On this example, the robot is facing toward the bottom of the Figure (note the two forward sensors with angles -15° and 15°). In this case, the computed distances are all greater than the sensor range: the robot is in empty space.

### B. Behaviors

Depending on the sensory situation, the robot applies different behaviors. We have defined three behaviors, implemented using deterministic rules (which we omit here for brevity), and defined their application conditions as follows.

The GO_STRAIGHT behavior simply displaces the robot in a straight line following the robot direction. This behavior is applied when the robot does not sense anything on its sensors (EMPTY_SPACE predicate).

The FOLLOW_WALL behavior displaces the robot parallel to the wall, if one is detected (NEAR_ONE_WALL predicate). This behavior exists in two versions, depending on the side the wall is detected. When the wall is first sensed on the left (resp. right), the robot keeps the wall on its left (resp. right). When this choice of side is made, it is kept, so that the robot does not do U-turns along a wall.

The FOLLOW_CORRIDOR behavior displaces the robot parallel to two walls, when they are detected on each side of the robot (NEAR_TWO_WALLS predicate).

Alg. IV.1 summarizes the method used for simulating the robot.

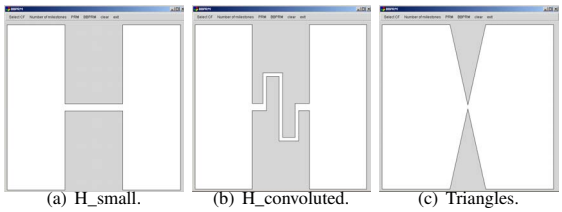
### C. Roadmap construction example

Fig. 3 shows one series of milestones and their links using BBPRM. In our implementation, the length of series $S$ was set to 50, $k$, the number of application of the behavior before putting a milestone, was set to 20, and $\tau$, the distance threshold for linking milestones using the classic local planner, was set to 5 units of distance.

On this example, the initial robot position was drawn on the right side of the configuration space. The robot was initially in empty space: the first few milestones are in a straight line, following the initial drawn direction (towards west-south-west). When the robot got near the wall, it followed it by

**ALG. IV.1** The SIMULATE_ROBOT subroutine. The input is an initial configuration `q`, and the output is the configuration obtained after simulating the behaviors.

```
repeat k times
 if EMPTY_SPACE(q) then
    q' ← GO_STRAIGHT(q)
    else
      if NEAR_ONE_WALL(q) then
          Decide left or right
          q' ← FOLLOW_WALL(q)
          else
             if NEAR_TWO_WALLS(q) then
                q' ← FOLLOW_CORRIDOR(q)
 q ← q'
return q
```



(a) H_small. (b) H_convoluted. (c) Triangles.

Fig. 4. Configuration spaces used for the experimental evaluation.

keeping it on its right side (moving south). Approaching the corridor, the robot entered it and followed it until getting out of it.

Because the milestones are in a single series, they are linked using only behavior links, which appear in green. One thing to notice is that the behavior links are shown as straight lines, even though these straight lines have no physical meaning. For instance, in the corridor, some of them seem to cross walls. The robot actually does not follow this trajectory, but moves according to a behavior, and does not collide with walls. The green segments are used only for showing the connectivity of the built roadmap.

## V. Experimental results

### A. Criterion

In order to validate the BBPRM approach, we have studied its ability to solve narrow passages. The criterion used was visually assessed: when the built roadmap connected the left and right parts of the configuration space, the narrow passage was said to be solved.

### B. Results

We have experimentally tested BBPRM on three different configuration spaces, which are shown Fig. 4. For each configuration space, the basic PRM has been implemented and tested 20 times using 1000 milestones, and BBPRM has been run 20 times with 150 milestones (3 series of $S = 50$ milestones).

The results are as follows:

- On the H_small configuration space (see Fig. 4(a)), PRM solved the narrow passage 25% of the time, and BBPRM 100% of the time.
- On the H_convoluted configuration space (see Fig. 4(b)), PRM solved the narrow passage 0% of the time, and BBPRM 55% of the time.
- On the Triangles configuration space (see Fig. 4(c)), PRM solved the narrow passage 60% of the time, and BBPRM 100% of the time.

On all these examples, BBPRM with 1000 milestones always succeeded (100% success). Fig. 5 shows an example of a roadmap build by BBPRM with 1000 milestones, which solves the H_convoluted narrow passage. In comparison, PRM with the same number of milestones on the same environment always fails (see Fig. 6).

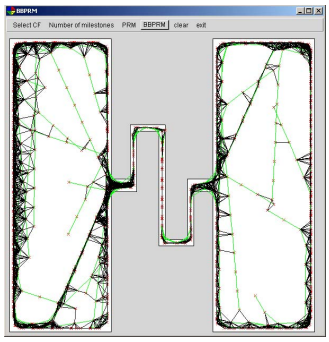


Fig. 5. A roadmap generated by BBPRM with 1000 milestones.

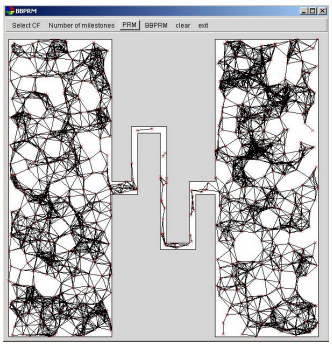


Fig. 6. A roadmap generated by PRM with 1000 milestones.

## VI. Discussion

### A. Computational cost

The computational cost of BBPRM is of the same order as PRM. The only additional cost is the one required by the application of the behaviors. To do this, one possible expensive component is the simulation of the sensors. In particular, they require the geometric computation of the distance to the obstacles, which gets more expensive as the number of polygon edges increases. In our implementation, this limits the complexity of the configuration spaces. For example, the running time of BBPRM with 1000 milestones on the H_convoluted example is around 1 minute. This could be greatly improved using distance computation techniques, like hierarchical bounding boxes for example. Moreover, if BBPRM was applied on a real robot physically exploring its environment, such computations would not be needed anymore, as physical sensors would be used.

### B. Probabilistic completeness

The BBPRM method is as probabilistically complete as the PRM method, since it relies on it for generating one milestone out of $S$, and it also uses the classic link method. This means that there is no configuration space which can be solved by PRM but not by BBPRM, if enough milestones are generated.

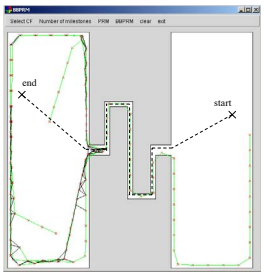



Fig. 7. An optimal path according to the travelled distance measure.

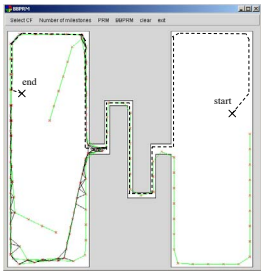



Fig. 8. Using the BBPRM roadmap may generate suboptimal paths.

### C. Generated trajectories

BBPRM suffers from a major drawback. When the roadmap is used for solving queries, it may generate trajectories which are suboptimal according to the travelled distance measure. This is the case because the milestones are mostly distributed along the obstacle boundaries. Typical trajectories therefore tend to keep the robot close to the walls. This drawback is the same for other PRM methods which try to generate milestones near the boundaries of obstacles. This effect is illustrated Fig. 7 and 8, which show respectively an optimal path according to the travelled distance and a suboptimal path generated by BBPRM.

However, when considering the life science point of view, the travelled distance measure is not the only cost function to take into account. For instance, staying close to obstacle boundaries brings more sensory information than wide empty space. This has already been exploited in the "coastal navigation" scheme [14]. In our case, paths generated by BBPRM already implicitly take into account this sensory information criterion.

## VII. Conclusion

We have shown the feasibility of the BBPRM approach as an alternative to the basic PRM approach. We have developed a new algorithm for generating probabilistic roadmaps that integrate behavior simulation, so as to better reflect the low-level sensorimotor capabilities of the robot. Furthermore, we have shown that BBPRM treats narrow passages more efficiently than the basic PRM.

We do not claim that our method could easily scale as is to high-dimensional spaces, for solving industrial robotics problems. Indeed, in a $n$ dimensional space, obstacle boundaries are $n-1$ dimensional. We would have to assume particularly ad hoc behaviors, that would especially find the holes in $n-1$ dimensional spaces. But, at the risk of belaboring our point, we do instead believe that our work validates the behavior-based approach for roadmap based path planning, as a possible candidate to hierarchical extensions of the PRM planning for the life sciences. In this context, sub-spaces at each level of the hierarchy are low-dimensional, and our approach is applicable and shows a possible model of the imbrication between low-level behaviors and path planning.

## Acknowledgment

E.S. wishes to thank gratefully Dr. David Hsu of the National University of Singapore (NUS), for advice and guidance. J.D. acknowledges A/P Marcelo. H. Ang Jr. and an INRIA international scholarship for support during his post-doctoral visit at NUS.

## References

[1] M. Ernst and M. Banks, "Humans integrate visual and haptic information in a statistically optimal fashion," *Nature*, vol. 415, no. 6870, pp. 429–33, 2002.

[2] J.-C. Latombe, *Robot Motion Planning*. Boston: Kluwer Academic Publishers, 1991.

[3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[4] E. Mazer, J.-M. Ahuactzin, and P. Bessière, "The Ariadne's clew algorithm," *Journal of Artificial Intelligence Research*, vol. 9, pp. 295–31, 1998.

[5] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *International Journal of Computational Geometry & Applications*, vol. 9, no. 4 & 5, pp. 495–512, 1999.

[6] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "Motion planning for a rigid body using random networks on the medial axis free space," in *Proceedings of the 15th Annual ACM Symposium on Computational Geometry*, 1999, pp. 173–180.

[7] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Proceedings of the Workshop on Algorithmic Foundations of Robotics (WAFR'98)*, 1998, pp. 155–168.

[8] V. Boor, M. H. Overmars, and A. F. van der Stappen, "Gaussian sampling for probabilistic roadmap planners," University of Utrecht, Tech. Rep. UU-CS-2001-36, 2001.

[9] D. Hsu, L. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *In Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998, pp. 141–154.

[10] F. Lingelbach, "Path planning using probabilistic cell decomposition," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA04)*, New Orleans, LA, USA, 2004, pp. 467–472.

[11] J. P. van den Berg and M. H. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA04)*, New Orleans, LA, USA, 2004, pp. 453–460.

[12] J. Burlet, O. Aycard, and T. Fraichard, "Robust motion planning using markov decision processes and quadtree decomposition," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA04)*, New Orleans, LA, USA, 2004, pp. 2820–2825.

[13] H. Kurniawati and D. Hsu, "Workspace importance sampling for probabilistic roadmap planning," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (to appear)*, 2004.

[14] N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal navigation - mobile robot navigation with uncertainty in dynamic environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA99)*, 1999, pp. 35–40.