Sampling Techniques for Probabilistic Roadmap Planners

Roland Geraerts Mark H. Overmars
Institute of Information and Computing Sciences
Utrecht University, the Netherlands
Email: {roland,markov}@cs.uu.nl.

Abstract. The probabilistic roadmap approach is a commonly used motion planning technique. A crucial ingredient of the approach is a sampling algorithm that samples the configuration space of the moving object for free configurations. Over the past decade many sampling techniques have been proposed. It is difficult to compare the different techniques because they were tested on different types of scenes, using different underlying libraries, implemented by different people on different machines. We compared 12 of such sampling techniques within a single environment on the same scenes. The results were surprising in the sense that techniques often performed differently than claimed by the designers. The study also showed how difficult it is to evaluate the quality of the techniques. The results should help users in deciding which technique is suitable for their situation.

1 Introduction

The motion planning problem asks for computing a collision-free, feasible motion for an object (or kinematic device) from a given start to a given goal placement in a workspace with obstacles. Besides the obvious application within robotics, motion planning also plays an important role in animation, virtual environments, computer games, computer aided design and maintenance, and computational chemistry.

Over the years, many different approaches to solving the motion planning problem have been suggested. See the book of Latombe[23] for an extensive overview of the situation up to 1991 and e.g. the proceedings of the yearly IEEE International Conference on Robotics and Automation (ICRA) or the Workshop on Foundations of Robotics (WAFR) for many more recent results. A popular motion planning technique is the *probabilistic roadmap planner (PRM)*, developed independently at different sites [3, 4, 19, 20, 26, 30]. It turns out to be very efficient, easy to implement, and applicable for many different types of motion planning problems (see e.g. [10, 14, 16, 21, 22, 27, 28, 30, 31, 32]).

The PRM approach consists of a preprocessing phase and a query phase. In the preprocessing phase a roadmap graph $G = (V, E)$ is constructed. In the query phase, the start and goal configurations are connected to the graph. The path is obtained by performing a Dijkstra's shortest path query on the graph. Let $\mathcal{C}$ denote the configuration space, that is, the space of all possible placements for the moving object. Let $\mathcal{C}_{\text{free}}$ denote the part of $\mathcal{C}$ that consists of feasible configurations (that is, without collisions and allowed for the moving object). We sample $\mathcal{C}_{\text{free}}$ for collision-free placements that are added as nodes to the graph G. Pairs of promising nodes are chosen in the graph and a simple local motion planner (normally a straight-line motion) is used to try to connect such placements with a path. If successful an edge is added to the graph. This process continues until the graph covers the connectedness of $\mathcal{C}_{\text{free}}$. The algorithm looks as follows.

Algorithm 1 CONSTRUCTROADMAP
```
1: loop
2:     $c \leftarrow$ a (useful) configuration in $\mathcal{C}_{\text{free}}$
3:     $V \leftarrow V \cup \{c\}$
4:     $N_c \leftarrow$ a set of (useful) nodes chosen from $V$
5:     for all $c' \in N_c$, in order of increasing distance from $c$ do
6:         if $c'$ and $c$ are not connected in $G$ then
7:             if the local planner finds a path between $c'$ and $c$ then
8:                 add the edge $c'c$ to $E$
```

The basic PRM approach leaves many details to be filled in, in particular how to sample the space, what local planner to use and how to select promising pairs. Over the past decade, researchers have investigated these aspects and developed many improvements over the basic scheme (see e.g. [1, 6, 7, 17, 21, 25, 27, 31, 33]). Unfortunately, the different improvements suggested are difficult to compare because they were tested on different types of scenes, using different underlying libraries, implemented by different people on different machines. In addition, the effectiveness of a technique sometimes depends on choices made for other parts of the method. So it is still rather unclear what is the best technique under which circumstance. (See [12] for a first study of this issue.)

In a previous paper [13] we made a first step toward a comparison between the different techniques developed. We implemented a number of the techniques in a single motion planning system and added software to compare the approaches. In this paper we further concentrate on the choice of sampling technique, comparing a large number of additional techniques on different scenes. This comparison gives insight in the relative merits of the techniques and the applicability in particular types of motion planning problems. The results are at times surprising in the sense that many of the claims of the creators could not be verified and sometimes our results even contradicted them.

The paper is organized as follows. In Section 2 we describe our experimental setup and the scenes we used. We also give some insight in the effects of other ingredients of the PRM approach, in particular collision checking and the choice of neighbor selection method. In Section 3 we study five different uniform sampling techniques. In Section 4 we consider seven non-uniform sampling techniques that have been designed to deal with the so-called *narrow passage* problem. Finally, in Section 5 we study the variation of the running time over different runs and present a simple technique with which this can be reduced.

2 Experimental Setup

In this study we restricted ourselves to free-flying objects in a three-dimensional workspace. Such objects have six degrees of freedom (three translational and three rotational). In all experiments, we used the most simple local method that consists of a straight-line motion in configuration space. For other types of devices or local planners the results might be different.

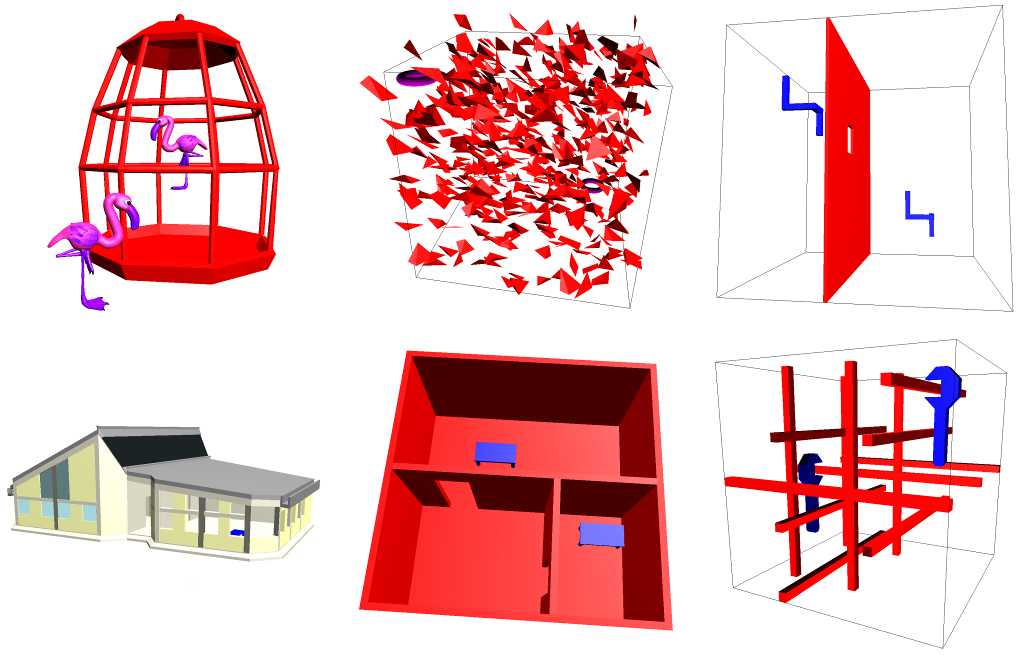

Figure 1: The 6 scenes used for testing

The PRM approach builds a roadmap which, in the query phase, is used for motion planning queries. We aim at computing a roadmap that covers the free space adequately but this is difficult to test. Instead, in each test scene we defined a relevant query and continued building the roadmap until the query configurations were in the same connected component.

All techniques were integrated in a single motion planning system called SAMPLE (System for Advanced Motion PLanning Experiments), implemented in Visual C++ under Windows XP. All experiments were run on a 2.66 GHz Pentium 4 processor with 1 GB internal memory. We used Solid as basic collision checking package [5]. In all experiments we report the running time in seconds. Because the experiments were conducted under the same circumstances, the running time is a good indication of the efficiency of the technique. For those techniques where there are random choices involved we report the average time over 30 runs.

For the experiments we used the following six scenes (see Figure 1). Note that these scenes are different from the ones in [13]. Futhermore, to make extensive experimentation possible, we did not include huge environments such as those common in CAD environments.

cage This environment consists of many primitives. The flamingo (7049 polygons) has to find a route (from a few dozen possibilities) that leads him out of the cage (1032 triangles). The complexity of this environment will put a heavy load on the collision checker but the paths are relatively easy.

clutter This scene consists of 500 uniformly distributed tetrahedra. A torus must move among them from one corner to the other. The configuration space will consist of many narrow corridors. There are many solutions to the query.

hole The moving object consists of four legs and must rotate in a complicated way to get through the hole. The hole is placed off-center to avoid that certain grid-based methods have an advantage. The configuration space will have two large open areas with two narrow winding passages between them.

house The house is a complicated scene consisting of about 2200 polygons. The moving

	collision checking			
	incremental	**binary**	**line**	**rotate-at-s**
cage	2.4	1.9	**1.8**	2.7
clutter	1.8	**1.3**	1.4	3.1
hole	431.9	**422.3**	436.4	1206.7
house	6.4	4.9	**4.7**	47.1
rooms	0.5	**0.4**	**0.4**	1.1
wrench	0.9	**0.5**	**0.5**	1.1

Table 1: Running times for different collision checking methods

object (table) is small compared to the house. Because walls are thin, the collision checker must make rather small steps along the paths, resulting in much higher collision checking times. Because of the many different parts in the scene the planner can be lucky or unlucky in finding the relevant part of the roadmap. So we expect a large difference in the running times of different runs.

rooms In this scene there are three rooms with lots of space and with narrow doors between them. So the density of obstacles is rather non-uniform. The table must move through the two narrow doors to the other room.

wrench This environment features a large moving obstacle (156 triangles) in a small workspace (48 triangles). There are many different solutions. At the start and goal the object is rather constrained.

Even though we concentrate on sampling in this paper, other aspects of the PRM approach have a big influence on the overall running time. As indicated in [13], the most time-consuming step in PRM is checking whether the motion produced by the local planner is collision free. There are different approaches for this. One can use an *incremental* approach that takes small steps along the path, or a *binary* approach that first checks the middle position of the path and then recurses on the two halves [27]. Also one can combine this with a *line* check to first see whether the origin of the moving object is collision free [11]. Finally, there is the *rotate-at-s* approach suggested in [2] that translates from start to an intermediary configuration s halfway, then rotates, and finally translates to the endpoint. Table 1 summarizes the results (using deterministic Halton points for sampling and a simple nearest-k node adding strategy; see below):

The table shows that in all scenes the binary approach was faster than the incremental approach although the improvement varied over the type of scene. The line check only had a marginal effect, contrary to the claim in [11]. Also the rotate-at-s technique did not give the improvement suggested in [2]. It should though be noticed that this can depend on the underlying collision checking package used. For the rest of the paper we will use the binary approach.

A second important choice to be made in PRM is how to select the set of neighbors to which we try to make connections. As each test for a connection is expensive we should try to avoid these as much as possible. On the other hand, if we try too few connections we will fail to connect the graph. It is not useful (from a complexity point of view) to make connections to nodes that are already in the same connected component, because a new connection will not help solving motion planning queries. Also, it is not useful to try to connect to nodes that lie too far away. The chance of success for such a connection is minimal while the collision checks required for testing the path are expensive.

We performed a large number of experiments to determine for each scene the optimal

	node adding strategy			
	nearest-k	**comp**	**comp-k**	**visibility**
cage	1.9	3.4	**1.6**	3.0
clutter	**1.3**	**1.3**	1.4	2.3
hole	409.5	7428.2	7554.4	**102.5**
house	4.9	**3.0**	13.0	45.5
rooms	0.4	**0.2**	**0.2**	6.3
wrench	0.5	**0.4**	**0.4**	1.6

Table 2: Comparison between 4 node adding strategies

values for the maximal distance d and for the maximum number of connections k. It turned out that for most test scenes a value of k between 20 and 25 was best. Only for the clutter scene a much smaller value of around 10 was required. The reason is that many connections are invalid due to the large number of obstacles. So even when there are many connected components it does not help to try to connect to them. For the maximal distance a similar argument holds. For large open scenes, like the cage and the wrench, a large value is best. For more constrained scenes, a smaller value is required. For the hole scene an even smaller value works best. The reason is that in the difficult part of the scene only short connections have a chance of success. We used the optimal values in the rest of this paper. In general, one would like to have a technique that determines the best values based on (local) properties of the scene. Such a technique is currently lacking.

As connected components play an important role, different techniques have been suggested to favor connections to components. While the standard *nearest-k* method tries to connect to the nearest k nodes, the *component* method tries to connect the new configuration to the nearest node in each connected component that lies close enough. The *component-k* method is a combination and tries to connect to at most k nodes in each connected component. The *visibility* sampling technique described in [25] is also a kind of connection technique. It tries to connect the new configuration to useful nodes. Usefulness is determined as follows: When a new node can be connected to no other nodes it forms a new connected component and is labeled useful. If it connects two or more components, it is also labeled useful. If it can be connected to just one component it is not labeled useful and removed.

Table 2 summarizes the results and confirms our earlier results in [13]. Although the visibility approach pruned the graph a lot, it still performed worse on most scenes. We feel that the reason is that the approach is too strict in rejecting nodes. Even though the component based techniques were often faster we used the nearest-k approach in the rest of the paper to avoid problems with the hole scene.

3 Uniform Sampling

The first papers on PRM used uniform random sampling of the configuration space to select the nodes that are added to the graph. In recent years, other uniform sampling approaches have been suggested to remedy certain disadvantages of the random behavior. In particular we will study the following techniques:

random In the random approach a sample is created by choosing random values for all its degrees of freedom. The sample is added when it is collision-free.

grid In this approach we choose samples on a grid. Because the grid resolution is unknown in advance, we start with a coarse grid and refine this grid in the process, halving the

	basic sampling strategy					
	random	**grid**	**halton**	**halton***	**rnd halton**	**cell-based**
cage	2.3	3.2	1.9	2.0	**1.5**	2.8
clutter	**1.2**	3.4	1.3	1.5	1.5	1.4
hole	433.9	370.4	422.3	**201.4**	435.5	279.5
house	78.7	**3.2**	4.9	10.4	17.9	10.0
rooms	0.7	0.8	**0.4**	0.7	0.6	0.7
wrench	0.7	**0.4**	0.5	**0.4**	0.7	**0.4**

Table 3: Comparison of average running times of 6 basic sampling strategies

cell size. Grid points on the same level of the hierarchy are added in random order.

halton In [8] it has been suggested to use so-called Halton point sets as samples. Halton point sets have been used in discrepancy theory to obtain a coverage of a region that is better than using a grid (see e.g. [9]). It has been suggested in [8] that this deterministic method is well suited for PRM.

halton* In this variant of halton we choose a random initial seed instead of setting the seed to zero [34]. The claims in [8] should still hold because they are independent of the seed.

random halton In this approach we use again halton points. But when adding the nth sample point we choose an area around this point (in configuration space) and choose a random configuration in this area. As size of the area we choose kA/n where A is the area of (the relevant part of) the configuration space and k is a small constant. So the area becomes smaller when more and more points are added. This added randomness should solve cases in which halton is unlucky.

cell-based In this approach we take random configurations within cells of decreasing size in the workspace. The first sample is generated randomly in the whole space. Next we split the workspace in 2^3 equally sized cells. In a random order we generate a configuration in each cell. Next we split each cell into sub-cells and repeat this for each sub-cell. This should lead to a much better spread of the samples over the configuration space.

An important question is how to choose random values. Random values were obtained by the Mersenne Twister [24]. Sampling for the rotational degrees of freedom ($SO3$) was performed by chosing random quaternions [29], except for halton based sampling techniques: we converted the (three Euler angle) halton values to a quaternion. See [35] for a more extensive elaboration on sampling methods for $SO3$.

Table 3 summarizes the results. It can be seen that the results were rather varying. In general the differences were rather small, that is, the slowest method took about twice the time of the fastest method. But for each scene another technique was the fastest. There was just one exception. The random approach performed reasonable well except in the house scene. For this particular scene there was a huge variation in running times between different runs. More that a third of the runs took 6 seconds or less, while another third took more than 100 seconds (one run even took 650 seconds). As a result, conclusions based on average times are difficult to make. Also the other approaches had a high variation in running time. Figure 2 shows a boxplot of the different running times for the rooms scene for the methods. The cell-based approach seemed to have the lowest variance.

It is interesting to compare halton and halton*. In some sense halton is simply one particular run of halton*. We saw that e.g. for the house scene it was a lucky run while for the hole scene it was an unlucky run. Adding a bit of randomness did not seem to remedy this problem, although our earlier results [13] seemed to suggest this.

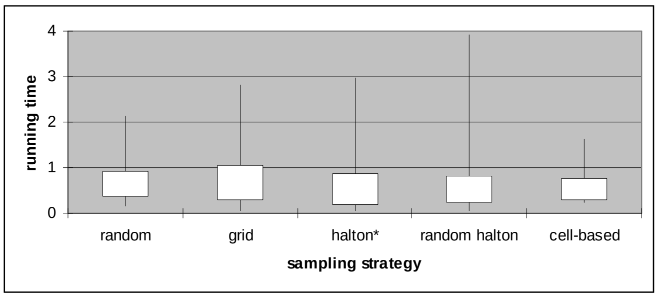

Figure 2: Box plot for the rooms scene, showing the variation in running time for 30 runs

In general we must conclude that there is little to win when using different kinds of uniform sampling in terms of average running time. There can though be other arguments to use a particular technique. For example, the halton approach is deterministic. Even though it might be unlucky on a particular scene this still is a favorable property.

4 Advanced Sampling

Rather than using uniform sampling, it has been suggested to add more samples in difficult regions of the environment. In this section we study a number of these techniques:

gaussian Gaussian sampling is meant to add more samples near obstacles. The idea is to take two random samples, where the distance between the samples is chosen according to a Gaussian distribution. Only if one of the samples lies in $\mathcal{C}_{\text{free}}$ and the other lies in $\mathcal{C}_{\text{forb}}$ do we add the free sample. This leads to a favorable sample distribution [7].

obstacle based This technique, based on [1], has a similar goal. We pick a random sample. If it lies in $\mathcal{C}_{\text{free}}$ we add it to the graph. Otherwise, we pick a random direction and move the sample in that direction with increasing steps until it becomes free and add the free sample.

obstacle based* This is a variation of the previous technique where we throw away a sample if it initially lies in $\mathcal{C}_{\text{free}}$. This will avoid many samples in large open regions.

bridge test The bridge test [15] is a hybrid technique that aims at better coverage of the free space. The idea is to take two random samples, where the distance between the samples is chosen according to a Gaussian distribution. Only if *both* samples lie in $\mathcal{C}_{\text{forb}}$ and the point in the middle of them lies in $\mathcal{C}_{\text{free}}$ the free sample is added. (To also get points in open space, every sixth sample is simply chosen random.)

medial axis This technique generates samples near the medial axis (MA) of the free space [33]. All samples have 2-equidistant nearest points resulting in a large clearance from obstacles. The method increases the number of samples in small volume corridors but is relatively expensive to compute.

nearest contact This method generates samples on the boundary of the C-space and can be seen as the opposite to the medial axis technique. First we choose a uniform random sample c. If c lies in $\mathcal{C}_{\text{free}}$ we discard it, else we calculate the penetration vector v between c and the environment. Then we move c in the opposite direction of v and place c on the boundary of the C-space. Care must be taken not to place c exactly on the boundary, because then it would be difficult to make connections between the samples.

We expect these techniques to be useful only in scenes where there are large open areas (in configuration space) and some narrow passages. Table 4 shows the results. (The halton* approach is shown for comparison.)

	sampling around obstacles						
	gaussian	**obstacle**	**obstacle***	**bridge**	**MA**	**nearest cont.**	**halton***
cage	7.3	3.1	5.3	3.3	215.9	5.4	**2.0**
clutter	2.8	2.0	2.6	3.3	620.4	7.1	**1.5**
hole	8.8	47.5	7.1	35.4	7.7	**2.3**	201.4
house	18.0	20.7	13.0	28.8	199.3	15.7	**10.4**
rooms	0.5	**0.4**	0.5	1.0	3.5	**0.4**	0.7
wrench	2.7	0.9	1.9	0.7	11.0	3.8	**0.4**

Table 4: Comparison of average running times of 7 advanced sampling strategies

As expected the techniques only performed considerably better for the hole scene. Also for the rooms scene the methods worked well but the improvement was not significant. However, in other situations the methods were up to 10 times slower. The medial axis approach was even worse, due to the expensive calculations. The method does though give samples that are nicely located between the obstacles which leads to motions with a higher clearance.

We conclude that special non-uniform techniques should only be used in specific situation with narrow corridors. Preferably they should also only be used in the parts of the workspace where this is relevant.

5 Variation of running times

An advantage of random sampling techniques is that they are usually fast and can succesfully deal with the diversity of problems. However, a price has to be paid: the running times needed to find a solution will vary. We noticed that some of them were exorbitantly high, increasing the average considerably. This phenomenon is undesirable because of two reasons. First, a large variation complicates statistical analysis and can even make it unreliable. Second, it is undesirable from a users point of view, e.g. in a virtual environment where real-time behavior is required, only a particular amount of cpu time will be scheduled for the motion planner.

A first study of this problem has been reported in [18], where bidirectional A^* search is used, based on parameterized formulas for increasing the competence of the local planner. We propose a new simple technique that can be used to decrease the maximum, average and variance of running times:

restart prm We run PRM for a particular time t. If no solution is found within t seconds, the PRM is restarted, throwing away the roadmap created so far. We repeat this process until a solution is found. Clearly, t should be a reasonable guess for the time in which we expect that the problem can be solved. If no such guess can be made we can also start with a small value and double the time t in each step.

As an example we apply the approach to the wrench scene. To make sure our averages make sense we ran the planner 2000 times. Figure 3 shows the percentage of runs that have been solved in a particular amount of time. We only show the tail of the chart because the heads of the two curves are indistinguishable.

Only 1% of the runs that were generated without restarting had a high running time (between 8.0 and 60.8 seconds). By restarting the PRM after 4 seconds this interval was dramatically improved to $[4.9; 8.2]$. This improvement positively changed the average, maximum and standard deviation of running times, which are stated in table 5. We conclude that restarting the PRM makes random techniques more robust. We plan to investigate this approach further.

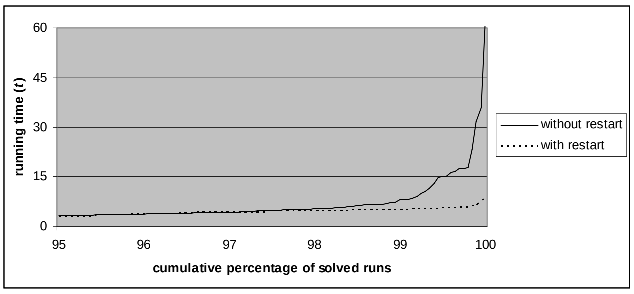

Figure 3: Cumulative percentages of 2000 runs solved in t seconds for the wrench scene.

	statistics for the variance problem			
	minimum	**average**	**maximum**	**st. deviation**
without restart	0.03	1.01	60.81	2.32
with restart	0.05	0.88	8.20	1.00

Table 5: Average running times for the wrench scene with and without restarting the PRM

6 Conclusions

In this paper we presented the results of a comparative study of various sampling techniques for the PRM approach to motion planning. The results showed that many claims on efficiency of certain sampling approaches could not be verified. The study also showed the difficult of evaluating the quality of the techniques. In particular the variance in the running time and the influence of certain bad runs are surprisingly large. We presented a very simple variance killer that seems to be effective. We are fairly though certain that better techniques exist. This is an interesting topic for further study. This study does not provide a final answer as to the best technique. Further research, in particular into adaptive sampling techniques, will be required for this.

Acknowlegements This work was supported by the Netherlands Organization for Scientific Research (N.W.O.) and by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2001-39250 (MOVIE - Motion Planning in Virtual Environments). We would like to thank Leon Bouquiet and Tim Schlechter who did a preliminary study on the variance of PRM, and Dennis Nieuwenhuisen who created part of the SAMPLE software.

References

[1] N. Amato, O. Bayazit, L. Dale, C. Jones, D. Vallejo, OBPRM: An obstacle-based PRM for 3D workspaces, in: P.K. Agarwal, L.E. Kavraki, M.T. Mason (eds.), *Robotics: The algorithmic perspective*, A.K. Peters, Natick, 1998, pp. 155–168.

[2] N. Amato, O. Bayazit, L. Dale, C. Jones, D. Vallejo, Choosing good distance metrics and local planners for probabilistic roadmap methods, *Proc. IEEE Int. Conf. on Robotics and Automation*, 1998, pp. 630–637.

[3] N. Amato, Y. Wu, A randomized roadmap method for path and manipulation planning, *Proc. IEEE Int. Conf. on Robotics and Automation*, 1996, pp. 113–120.

[4] J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, P. Raghavan, A random sampling scheme for path planning, *Int. Journal of Robotics Research* **16** (1997), pp. 759–774.

[5] G. van den Bergen, *Collision detection in interactive 3D computer animation*, PhD thesis, Eindhoven University, 1999.

[6] R. Bohlin, L.E. Kavraki, Path planning using lazy PRM, *Proc. IEEE Int. Conf. on Robotics and Automation*, 2000, pp. 521–528.

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

[8] M. Branicky, S. LaValle, K. Olson, L. Yang, Quasi-randomized path planning, *Proc. IEEE Int. Conf. on Robotics and Automation*, 2001, pp. 1481–1487.

[9] B. Chazelle, *The discrepancy method*, Cambridge University Press, Cambridge, 2000.

[10] J. Cortes, T. Simeon, J.P. Laumond, A random loop generator for planning the motions of closed kinematic chains using PRM methods, *Proc. IEEE Int. Conf. on Robotics and Automation*, 2002, pp. 2141–2146.

[11] L. Dale, *Optimization techniques for probabilistic roadmaps*, PhD thesis, Texas A&M University, 2000.

[12] L. Dale, N. Amato, Probabilistic roadmaps – Putting it all together, *Proc. IEEE Int. Conf. on Robotics and Automation*, 2001, pp. 1940–1947.

[13] R. Geraerts, M.H. Overmars. A Comparative Study of Probabilistic Roadmap Planners. *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2002, pp. 43–57.

[14] L. Han, N. Amato, A kinematics-based probabilistic roadmap method for closed chain systems, *Proc. Workshop on Algorithmic Foundations of Robotics (WAFR'00)*, 2000, pp. 233–246.

[15] D. Hsu, T. Jiang, J. Reif, Z. Sun. The Bridge Test for Sampling Narrow passages with Probabilistic Roadmap Planners. *IEEE International Conference on Robotics & Automation*, 2003.

[16] C. Holleman, L. Kavraki, J. Warren, Planning paths for a flexible surface patch, *Proc. IEEE Int. Conf. on Robotics and Automation*, 1998, pp. 21–26.

[17] D. Hsu, L. Kavraki, J.C. Latombe, R. Motwani, S. Sorkin, On finding narrow passages with probabilistic roadmap planners, in: P.K. Agarwal, L.E. Kavraki, M.T. Mason (eds.), *Robotics: The algorithmic perspective*, A.K. Peters, Natick, 1998, pp. 141–154.

[18] P. Isto, M. Mäntylä, J. Tuominen, On Addressing the Run-Cost Variance in Randomized Motion Planners, *Proc. IEEE Int. Conf. on Robotics and Automation*, 2003, pp. 2934–2939.

[19] L. Kavraki, *Random networks in configuration space for fast path planning*, PhD thesis, Stanford University, 1995.

[20] L. Kavraki, J.C. Latombe, Randomized preprocessing of configuration space for fast path planning, *Proc. IEEE Int. Conf. on Robotics and Automation*, 1994, pp. 2138–2145.

[21] L. Kavraki, P. Švestka, J-C. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. on Robotics and Automation* **12** (1996), pp. 566–580.

[22] F. Lamiraux, L.E. Kavraki, Planning paths for elastic objects under manipulation constraints, *Int. Journal of Robotics Research* **20** (2001), pp. 188–208.

[23] J-C. Latombe, *Robot motion planning*, Kluwer Academic Publishers, Boston, 1991.

[24] M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator, *Trans. on Modeling and Comp. Simulation* Vol. **8**, No. 1, (1998), pp. 3–30.

[25] C. Nissoux, T. Siméon, J.-P. Laumond, Visibility based probabilistic roadmaps, *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, 1999, pp. 1316–1321.

[26] M.H. Overmars, *A random approach to motion planning*, Technical Report RUU-CS-92-32, Dept. Comput. Sci., Utrecht Univ., Utrecht, the Netherlands, October 1992.

[27] G. Sánchez, J.-C. Latombe, A single-query bi-directional probabilistic roadmap planner with lazy collision checking, *Int. Symposium on Robotics Research (ISRR'01)*, 2001.

[28] T. Simeon, J. Cortes, A. Sahbani, J.P. Laumond, A manipulation planner for pick and place operations under continuous grasps and placements, *Proc. IEEE Int. Conf. on Robotics and Automation*, 2002, pp. 2022–2027.

[29] K. Shoemake. Uniform random rotations. In *Graphics Gems III*, Academic Press, 1992, pp. 124–132.

[30] P. Švestka, *Robot motion planning using probabilistic roadmaps*, PhD thesis, Utrecht Univ. 1997.

[31] P. Švestka, M.H. Overmars, Motion planning for car-like robots, a probabilistic learning approach, *Int. Journal of Robotics Research* **16** (1997), pp. 119–143.

[32] P. Švestka, M.H. Overmars, Coordinated path planning for multiple robots, *Robotics and Autonomous Systems* **23** (1998), pp. 125–152.

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

[34] X. Wang. F,J, Hickernell. Randomized Halton Sequences. *Math. Comput. Model.* **32** (2000), pp. 887–899.

[35] A. Yershova. S.M. Lavalle. Deterministic Sampling methods for Spheres and $SO(3)$. Submitted to *Proc. IEEE Int. Conf. on Robotics and Automation*, 2004.