A Comparative Study of Probabilistic Roadmap Planners

Roland Geraerts Mark H. Overmars

Institute of Information and Computing Sciences, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands.
Email: [roland,markov]@cs.uu.nl.

Abstract. The probabilistic roadmap approach is one of the leading motion planning techniques. Over the past eight years the technique has been studied by many different researchers. This has led to a large number of variants of the approach, each with its own merits. 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. In this paper we provide a comparative study of a number of these techniques, all implemented in a single system and run on the same test scenes and on the same computer. In particular we compare collision checking techniques, basic sampling techniques, and node adding techniques. The results should help future users of the probabilistic roadmap planning approach to choose the correct techniques.

1 Introduction

The basic 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. Originally the problem was studied within the robotics community, but in recent years many new applications arise in such areas as 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[21] 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.

The *probabilistic roadmap planner (PRM)* is a relatively new approach to motion planning, developed independently at different sites [3,4,17,18,23,28]. It turns out to be very efficient, easy to implement, and applicable for many different types of motion planning problems (see e.g. [10,13,15,19,20,26,24,27–30]).

Globally speaking, the PRM approach samples the configuration space (that is, the space of all possible placements for the moving object) for collision-free placements. These are added as nodes to a roadmap graph.

Pairs of promising nodes are chosen in the graph and a simple local motion planner is used to try to connect such placements with a path. This process continues until the graph covers the connectedness of the space.

The basic PRM approach leaves many details to be filled in, like how to sample the space, what local planner to use and how to select promising pairs. Over the past eight years researchers have investigated these aspects and developed many improvements over the basic scheme (see e.g. [1,6,7,16,19,22,24,29,31]). Unfortunately, the different improvements suggested are difficult to compare. Each author used his or her own implementation of PRM and used different test scenes, both in terms of environment and the type of moving device used. Also the effectiveness of one 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 circumstances. (See [12] for a first study of this issue.)

In this paper we make a first step toward a comparison between the different techniques developed. We implemented a large number of the techniques in a single motion planning system and added software to compare the approaches. In particular we concentrated on the sampling technique and the choice of promising pairs of nodes. This comparison gives insight in the relative merits of the techniques and the applicability in particular types of motion planning problems. Also we hope that in the longer run our results will lead to improved (combinations of) techniques and adaptive approaches that choose techniques based on observed scene properties.

The paper is organized as follows. In Section 2 we review the basic PRM approach. In Section 3 we describe our experimental setup and the scenes we use. In Section 4 we compare different ways of performing collision checks of the paths produced by the local planner. We conclude that a binary approach performs best. In Section 5 we consider different sampling strategies. We conclude that, except for very special cases, one best uses a deterministic approach based on Halton points, with a small amount of added randomness. In Section 6 we study different strategies for choosing promising pairs of nodes to connect. We conclude that one best picks a few nodes in each connected component of the roadmap. These results partially contradict earlier claims.

2 The PRM Method

The motion planning problem is normally formulated in terms of the *configuration space* $\mathcal{C}$, the space of all possible placements of the moving object. Each degree of freedom of the object corresponds to a dimension of the configuration space. Each obstacle in the workspace, in which the object moves, transforms into an obstacle in the configuration space. Together they form the forbidden part $\mathcal{C}_{\text{forb}}$ of the configuration space. A path for the moving object corresponds to a curve in the configuration space connecting the start and the goal configuration. A path is collision-free if the corresponding curve

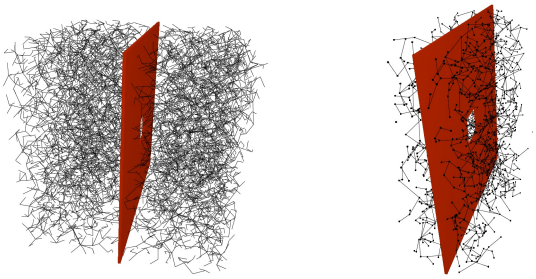

Fig. 1. The roadmap graph we get for the difficult hole test scene used in this paper. The left image shows the graph using halton sampling and the right image uses gaussian sampling.

does not intersect $\mathcal{C}_{\text{forb}}$, that is, it lies completely in the free part of the configuration space, denoted with $\mathcal{C}_{\text{free}}$.

The probabilistic roadmap planner samples the configuration space for free configurations and tries to connect these configurations into a roadmap of feasible motions. There are a number of versions of PRM, but they all use the same underlying concepts.

The global idea of PRM is to pick a collection of (useful) configurations in the free space $\mathcal{C}_{\text{free}}$. These free configurations form the nodes of a graph $G = (V, E)$. A number of (useful) pairs of nodes are chosen and a simple local motion planner is used to try to connect these configurations by a path. When the local planner succeeds an edge is added to the graph. The local planner must be very fast, but is allowed to fail on difficult instances. A typical choice is to use a simple interpolation between the two configurations, and then check whether the path is collision-free. So the path is a straight line in configuration space.

Once the graph reflects the connectivity of $\mathcal{C}_{\text{free}}$ it can be used to answer motion planning queries. (See Figure 1 for an example of the roadmap graphs computed.) To find a motion between a start configuration and a goal configuration, both are added to the graph using the local planner. (Some authors use more complicated techniques to connect the start and goal to the graph, e.g. using bouncing motion.) Then a path in the graph is found which corresponds to a motion for the object. The pseudo code for the algorithm for constructing the graph is shown in Algorithm ConstructRoadmap.

Note that in this version of PRM we only add an edge between nodes if they are not in the same connected component of the roadmap graph. This saves time because such a new edge will not help solving motion planning queries. On the other hand, to get short paths such extra edges are useful (see e.g. [14,25]). For this comparative study we will not add these additional edges.

In this study we concentrate on the various choices for picking useful samples, in line 2 of the algorithm, and for picking useful pairs of nodes for adding edges, that is, on the choice of N_c in line 4. These are the most crucial

Algorithm 1 ConstructRoadmap

Let: $V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
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

steps and they strongly influence the running time and the structure of the roadmap graph.

3 Experimental Setup

The probabilistic roadmap approach has been applied to many different types of motion planning problems involving different kinds of robotic devices. In this comparative study we restrict ourselves to free-flying objects in a three-dimensional workspace. Such objects have six degrees of freedom, three translational degrees and three rotation degrees. In a later study we will consider other devices.

In all experiments we use the most simple local method that consists of a straight-line motion in configuration space, interpolating between the start and goal configuration. In a later study we will consider the effect of using more sophisticated local planners.

An important distinction is between single shot methods and preprocessing methods. A single shot method asks to solve a particular motion planning query for a given start and a given goal configuration. A preprocessing method builds a data structure that can then be used to solve different motion planning queries efficiently. We feel that the PRM approach is particularly useful as a preprocessing method and, hence, we choose that method in this paper. This means that we won't consider techniques like bidirectional methods and lazy PRMs [6,24] which are particularly suited for single shot situations and situations were just a few queries occur. We aim at computing a roadmap that is checked for collisions and covers the free space adequately. To this end, in each scene (see below) we defined a relevant query that can only be solved when there is an adequate covering of the free space. We continue building the roadmap until the query configurations lie 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

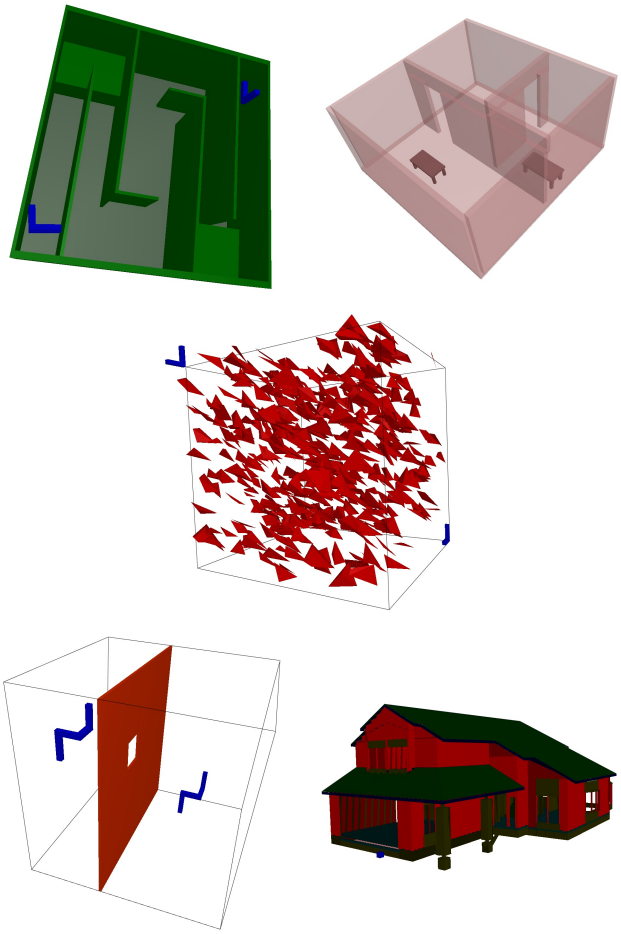

Fig. 2. The scenes used for testing, together with the start and goal configuration for the moving object.

a 2.4 GHz Pentium 4 processor with 1 GB of internal memory. 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 20 runs. When there is a large difference between runs we will indicate this.

For the experiments we used the following five scenes (see Figure 2):

corridor This is the most simple scene. An L-shaped object must translate and rotate through a winding corridor to the other end. Some obstacles force rotation of the object. We expect that all methods can solve this problem very quickly.

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. The table is rather complex because it has four cylindrical legs, leading to more time-consuming collision checking.

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

hole A rather famous test scene. 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 method have an advantage. The configuration space will have two large open areas with (most likely) two narrow winding passages between them.

house The house is a complicated scene consisting of 1600 polygons. It has many small rooms. As a result a large roadmap will be required to cover the free space. Also, because walls are thin, the collision checker must make rather small steps along the paths, resulting in much higher collision checking times.

4 Collision Checking

The most time-consuming steps in the probabilistic roadmap planner are the collision checks that are required to decide whether a sample lies in $\mathcal{C}_{\text{free}}$ and whether the motion produced by the local planner is collision free. In particular the second type of checks is time consuming. In this section we investigate some techniques for collision testing of the paths.

As basic collision checking package we use Solid[5]. The advantage of this package is that it considers objects like blocks, tetrahedra, spheres, and cylinders as solids rather than boundary representations. This avoids the generations of samples inside obstacles. Also it reduces the number of obstacles required to describe complicated scenes. Solid builds a data structure based on bounding boxes for fast query answering.

When testing a path for collisions we can use the following techniques:

incremental In the incremental method we take small steps along the path from start to goal. For each placement we check for collision with the scene.

binary In the binary method we start by checking the middle position along the path. If it is collision free we recurse on both halves of the path checking the middle positions there. In this way we continue until either a collision is found or the checked placements lie close enough together (again determined by a given step size).

In early papers on PRM the incremental method was used. Later papers suggest that the binary method works better[24]. The reason is that the middle position is the one that has the highest chance of not being collision free. This means that, when the path is not collision free, this collision will most likely be found earlier, that is, after fewer collision checks.

We tested both approaches. To make the results comparable we used a deterministic version of PRM using Halton points for sampling and a simple nearest-n node adding strategy (see Sections 5 and 6). As a result both version create exactly the same roadmap. The following table shows the results for the five test scenes:

collision checking		
	incremental	binary
corridor	0.5	**0.2**
rooms	0.8	**0.3**
clutter	8.7	**4.0**
hole	44.2	**43.3**
house	380.0	**207.4**

As can be seen from the table, in all scenes the binary approach is faster although the improvement varies over the type of scene. This can be explained from the fact that in the hole scene path checks are more often successful than in the other scenes. The binary approach only helps when the check fails.

It has been suggested that one should try to compute sweep volumes and use these for collision tests. As a result, a path check would require just one collision test. The problem is that it is very difficult to compute sweep volumes for three-dimensional moving objects with six degrees of freedom. A much simpler technique is to first check with the sweep volume introduced by the origin of the object, that is, with a line segment between start and goal position (see [11]).

line check In this method we first perform a collision test with the line segment between the start and goal position in the workspace. Only if it is collision-free we perform either the incremental or the binary method. (This assumes that the origin of the object lies inside it.)

We would expect that this test will quickly discard many paths that have a collision, leading to an improvement in running time. The following table summarizes the results:

binary collision checking		
	no line	line check
corridor	**0.2**	**0.2**
rooms	**0.3**	**0.3**
clutter	**4.0**	4.3
hole	**43.3**	44.5
house	**207.4**	208.1

As can be seen, against our expectation, the line check did not lead to any improvement. (It does help for the incremental method but the bounds are still worse than for the binary method.) Sometimes it is even slightly worse. We believe that this is due to the way Solid performs the collision checking with the line segment. The line segment will normally have a rather large bounding box. As a result the bounding box test will not be enough to discard many obstacles. This requires more precise tests, which are more expensive. Also, Solid will always report all intersections, not just the first one. As the line might intersect multiple obstacles this will lead to additional (redundant) work. Other collision checking packages might treat such line intersections differently. We will investigate this in the future. Also, in all test scenes, except for the rooms scene, our moving objects are rather simple. When the moving object gets more complex, collision tests with it will become more expensive. In such a case the line check might lead to an improvement. In [11] it was suggested to only apply the line check when the distance between the endpoints is large. We tried this but did not see any significant improvement in performance. We conclude that, with Solid, we best use binary collision checking without line checks. This is the technique we will use in the remainder of this paper.

5 Basic Sampling

The first papers on PRM used uniform random sampling of the configuration space to select the nodes to add to the graph. In recent years other approaches have been suggested, either to create more samples in difficult regions or to remedy certain disadvantages of the random behavior. We will first study and compare some of the second type of approaches. 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. This was the traditional way of creating samples.

grid It has often been suggested to simply choose samples on a grid. Because we don't know the required grid resolution in advance we start with a coarse grid and refine this grid in the process, halving the 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 good coverage of a region with points that is better than using a grid (see e.g. [9]). Even though the criteria in PRM are rather different than in discrepancy theory it has been suggested in [8] that the method is well suited for PRM. The method is deterministic!

cell-based In this approach we take random configuration 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 2^3 sub-cells and repeat this for each sub-cell, etc. The rationale behind this is that we get a much better spread of the positions over the configuration space. (Also choosing cells for the rotational degrees of freedom would lead to too many cells.)

Note that we do not include visibility sampling[22] in this test. We consider visibility sampling as a node adding technique rather than a sampling technique and, hence, it is described in the next section.

The following table summarizes the results.

	basic sampling strategy			
	random	grid	halton	cell-based
corridor	0.4	1.4	**0.2**	0.4
rooms	0.8	0.6	**0.3**	0.7
clutter	**2.7**	7.3	4.0	3.1
hole	33.9	**26.2**	43.3	42.3
house	210.0	262.5	**207.4**	225.9

It can be seen that the results are rather varying and that the deterministic halton approach is not always the fastest. In the hole scene it is even considerably slower. The numbers contradict the results reported in [8] were considerable improvements are reported. This might be due to the much smaller connection distance chosen in [8] which seems to favor the halton sampling. Such a smaller connection distance though increases the total time bound. A big advantage of the halton approach though is that the results are always the same while for the random approach and to a lesser extend for the cell-based approach there is a large variation in running time. A few bad samples added at the beginning can complicate the connection process considerably. It has been suggested that this can be remedied by from time to time restarting the process. We did not test this further. In general we were surprised by the small differences. It seems the sampling approach is not as critical as people have suggested before.

So we can conclude that the major advantage of using halton points does not lie in the faster running time but in the deterministic nature that guarantees that the running time is always the same. But at the same time this is

the weakness of the method. In the hole scene this can be seen. Here halton seems to be unlucky in the configurations it picks. For any deterministic sampling technique one can construct scenes for which it will take a huge amount of running time. We suggest to remedy this by adding a small amount of randomness to the method:

random halton In this approach we use 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 (in the order of 0.002 for the experiments we conducted). So the area becomes smaller when more and more points are added.

The following table shows the difference between halton and random halton.

sampling with halton points		
	deterministic	random
corridor	**0.2**	0.3
rooms	0.3	**0.2**
clutter	4.0	**2.2**
hole	43.3	**15.8**
house	207.4	**152.9**

We conclude that adding a little bit of randomness does improve the running time in almost all cases. The approach is better than any of the other sampling techniques considered. So we recommend this method to be used. (In the next section we do use halton sampling without randomness. The reason is that this is easier for testing because we do not have to average over different runs.)

As indicated above other sampling techniques have been suggested that try to add more samples in difficult regions of the environment. It has for example been suggested to add more samples near to obstacle boundaries[1,7], to allow samples inside obstacles and push the to the free space[1,16] and to sample near to the Voronoi diagram of the obstacles[31]. Here we compare three 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 the $\mathcal{C}_{\text{free}}$ and the other lies in $\mathcal{C}_{\text{forb}}$ do we add the free sample. It has been shown that 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 if 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 (or lies outside the scene). We add the free sample. This will normally lie close to an obstacle boundary.

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.

We expect these techniques to be useful only in scenes where there are large open areas (in configuration space) and some narrow passages. The results are as follows, comparing them to the random halton approach (see also Figure 1).

	sampling around obstacles			
	gaussian	obstacle	obstacle*	halton
corridor	0.4	0.5	0.5	**0.3**
rooms	0.4	0.7	0.5	**0.2**
clutter	5.6	3.7	7.9	**2.2**
hole	3.1	8.0	**2.1**	15.8
house	268.0	197.1	209.4	**152.9**

As expected the techniques only perform better than halton for the hole scene. As the techniques can be quite a bit worse for other scenes they cannot be used as a general approach. Because all three techniques use a lot of randomness they suffer from the same problem as random sampling that some runs are very bad. We are currently investigating combinations of these techniques with halton sampling.

6 Node Adding

In this section we will study the second important choice to be made in PRM: 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. Connected components in the graph play a crucial role here. We like to connect such components into larger components whenever we can. Also, we need to create new components in unexplored parts of the configuration space.

There are two global observations that can be made. First of all, as indicated in Section 2, it is not useful (from a complexity point of view) to make connections to nodes that are already in the same connected component. Secondly, it is never useful 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. An important question here is what we use as a distance measure (see also [2]). In particular, it is unclear

how to take the rotational distance into account. In this paper we use $d_t + d_r$ where d_t denotes the translational distance of the origin of the object and d_r denotes the distance traveled by the point of the object furthest from its origin, while performing the rotation. This measure is easy to compute and gives a reasonably estimate.

We consider the following techniques:

nearest-n Here we try to connect the new configuration c to the nearest n nodes in the graph. The rationale is that nearby nodes lead to short connections that can be checked efficiently. If many nearby nodes lie in the same connected component there is no other component in the neighborhood so it is acceptable that we only connect to a single component.

component Here we try to connect the new configuration to the nearest node in each connected component that lies close enough. The rationale is that we prefer to connect to multiple connected components.

component-n Here we try to connect the new configuration to at most n nodes in each connected component. Still we keep the total number of connections tried small (the same number as for nearest-n). The rationale is that when the number of components is small we prefer to spend some extra time on trying to make connections. Otherwise the time required for adding the node will become the dominant factor.

visibility This method is based on the visibility sampling technique described in [22]. We try 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. (In [22] useless nodes are deleted but there is actually no reason for this. They won't take any time and they can be useful in query answering.) It has been observed in [22] that the number of useful nodes remains small, making it possible to try connections to all of them.

The following table summarizes our test results. In all tests we use deterministic halton sampling. As maximal connection distance we use a box that is about half the size in each dimension of the configuration space. For the nearest-n method we picked a value for n of about 15 (depending a bit on the scene). For component-n we used a value of 3 in the experiments.

	node adding strategy			
	nearest-n	comp	comp-n	visibility
corridor	**0.2**	0.8	0.5	1.5
rooms	**0.3**	0.5	0.7	2.3
clutter	4.0	4.4	**3.4**	7.6
hole	43.3	> 120	31.9	**20.9**
house	207.4	279.6	**189.1**	> 600

We were disappointed with the results for visibility adding. Only for the hole scene did it perform better (but here the obstacle based sampling techniques performed even better). From previous papers we had expected the method to perform better. We believe that this is due to the fact that other papers compare it with a method that is worse than nearest-n or component-n. As can be seen, the results are a bit varying. But the component-n method is in most cases the fastest and in the other cases almost the fastest. We conjecture that the reason is that it combines the following important aspects

- It aims at connecting different components.
- It never tries too many connections.
- It never tries too few connections.

The other methods lack in at least one of these aspects.

We plan to investigate node adding strategies further over the coming months. We believe that a better definition of the notion of useful nodes is required to improve the performance. Such a notion should probably change during the construction process and might also depend on local scene properties.

7 Conclusions

In this paper we presented the results of a comparative study of various PRM techniques. The results confirm previously made claims that the binary approach for collision checking works well. Previous claims on the quality of halton points could not be confirmed but a variation of halton, adding some randomness, gave a better performance in most scenes. For node adding it turned out that visibility sampling did not perform as well as expected. A technique based on choosing a number of nodes per component seemed to perform best. We plan to further investigate the issue of selecting promising nodes for connections. Also we plan to experiment with a number of additional techniques, in particular techniques aimed at narrow passages.

One thing that is clear from the study is that a careful choice of techniques is important. Also, it is not necessarily true that a combination of good techniques is good. For example, for the hole scene one might expect that a combination of obstacle based and visibility works best. But experiments show that this combination is actually about twice as bad (4.5 seconds).

We hope that our studies shed some more light on the questions what technique to use in which situation. A major challenge is to create planners that automatically choose the correct combination of techniques based on scene properties or that learn the correct settings while running.

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. 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.
14. O. Hofstra, D. Nieuwenhuisen, M.H. Overmars, Improving the path quality for probabilistic roadmap planners, to appear.
15. 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.
16. 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.
17. L. Kavraki, *Random networks in configuration space for fast path planning,* PhD thesis, Stanford University, 1995.
18. 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.
19. 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.
20. F. Lamiraux, L.E. Kavraki, Planning paths for elastic objects under manipulation constraints, *Int. Journal of Robotics Research* **20** (2001), pp. 188–208.

21. J-C. Latombe, *Robot motion planning*, Kluwer Academic Publishers, Boston, 1991.
22. C. Nissoux, T. Siméon, J.-P. Laumond, Visibility based probabilistic roadmaps, *Proc. IEEE Int. Conf. on Intelligent Robots and Systems,* 1999, pp. 1316–1321.
23. 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.
24. G. Sánchez, J.-C. Latombe, A single-query bi-directional probabilistic roadmap planner with lazy collision checking, *Int. Journal of Robotics Research*, 2002, to appear.
25. E. Schmitzberger, Probabilistic approach to list all non homotopic solutions of a motion planning problem, unpublished notes, 2002.
26. S. Sekhavat, P. Švestka, J.-P. Laumond, M.H. Overmars, Multilevel path planning for nonholonomic robots using semiholonomic subsystems, *Int. Journal of Robotics Research* **17** (1998), pp. 840–857.
27. 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.
28. P. Švestka, *Robot motion planning using probabilistic roadmaps,* PhD thesis, Utrecht Univ. 1997.
29. 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.
30. P. Švestka, M.H. Overmars, Coordinated path planning for multiple robots, *Robotics and Autonomous Systems* **23** (1998), pp. 125–152.
31. 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.