Available online at www.sciencedirect.com
SCIENCE DIRECT
Robotics and Autonomous Systems 54 (2006) 165–173
Robotics and Autonomous Systems
www.elsevier.com/locate/robot

Sampling and node adding in probabilistic roadmap planners

Roland Geraerts*, Mark H. Overmars

Utrecht University, Institute of Information and Computing Sciences, P.O. Box 80.089, 3584CH Utrecht, The Netherlands

Received 30 September 2004; accepted 13 September 2005
Available online 1 December 2005

Abstract

The probabilistic roadmap approach is one of the leading motion planning techniques. Over the past decade 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, sampling techniques, and node adding techniques. 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 future users of the probabilistic roadmap planning approach in deciding which technique is suitable for their situation.

Keywords: Comparative study; PRM; Motion planning; Sampling techniques; Node adding techniques

1. Introduction

Motion planning can be defined as finding a path for an object (or kinematic device) from a given start to a given goal placement in a workspace without colliding with obstacles in the workspace. 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 [26].

Over the years, many different approaches to solving the motion planning problem have been suggested. See the books of Latombe [25] and LaValle [27] for an extensive overview of the situation and for example the proceedings of the yearly IEEE International Conference on Robotics and Automation (ICRA) or the Workshop on Foundations of Robotics (WAFR) for many recent results. A popular motion planning technique is the Probabilistic Roadmap Method (PRM), developed independently at different sites [3,4,19,20,31,36]. It turns out to be very efficient, easy to implement, and applicable for many different types of motion planning problems (see e.g. [9,13,14, 21,24,33,35,36,38,39]).

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. If successful, an edge is added between the nodes in the graph. This process continues until the graph represents 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 decade researchers have investigated these aspects and developed many improvements over the basic scheme (see e.g. [2,6,5, 17,21,30,33,38,40]). 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 [11] for a first study of this issue.)

In [12], we made a first step toward a comparison between the different techniques developed. In this work, we will

* Corresponding author.
E-mail address: roland@cs.uu.nl (R. Geraerts).

0921-8890/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2005.09.026

continue comparing them. 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 approaches checking local paths for collisions, 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 term 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 six different uniform sampling strategies. We conclude that, except for very special cases, one best uses a deterministic approach based on Halton points, although the differences between the methods are small. In Section 6 we consider six non-uniform sampling techniques that have been designed to deal with the so-called *narrow passage* problem and conclude that these techniques should only be used in the parts of the workspace containing narrow passages, i.e., they do handle the test scene with one very narrow passage faster than uniform sampling but are considerable slower on all other scenes. In Section 7 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. Finally, in Section 8 we study the variation of the running time over different runs and present a simple technique that can be used to reduce this variation.

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 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 PRMs, 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

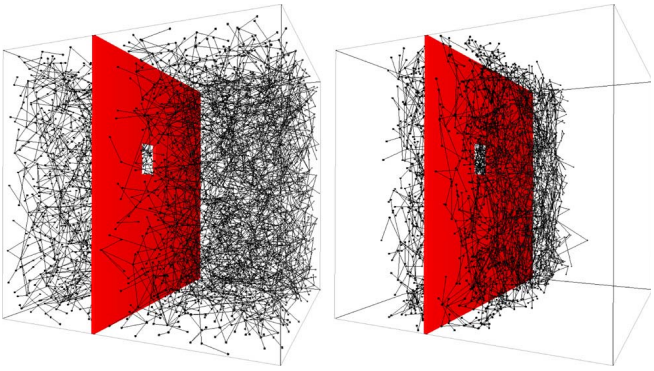


Fig. 1. The roadmap graph we get for the difficult hole test scene used in this paper. The left image shows the graph using Halton sampling and the right image uses Gaussian sampling.

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 Fig. 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 the Algorithm CONSTRUCTROADMAP. Note that in

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

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. [29]). For this comparative study we will not add these additional edges.

In this study we concentrate on the various choices for picking useful samples (line 2 of the algorithm), for picking useful pairs of nodes for adding edges (that is, on the choice of N_c in line 4) and for collision checking those edges (line 7).

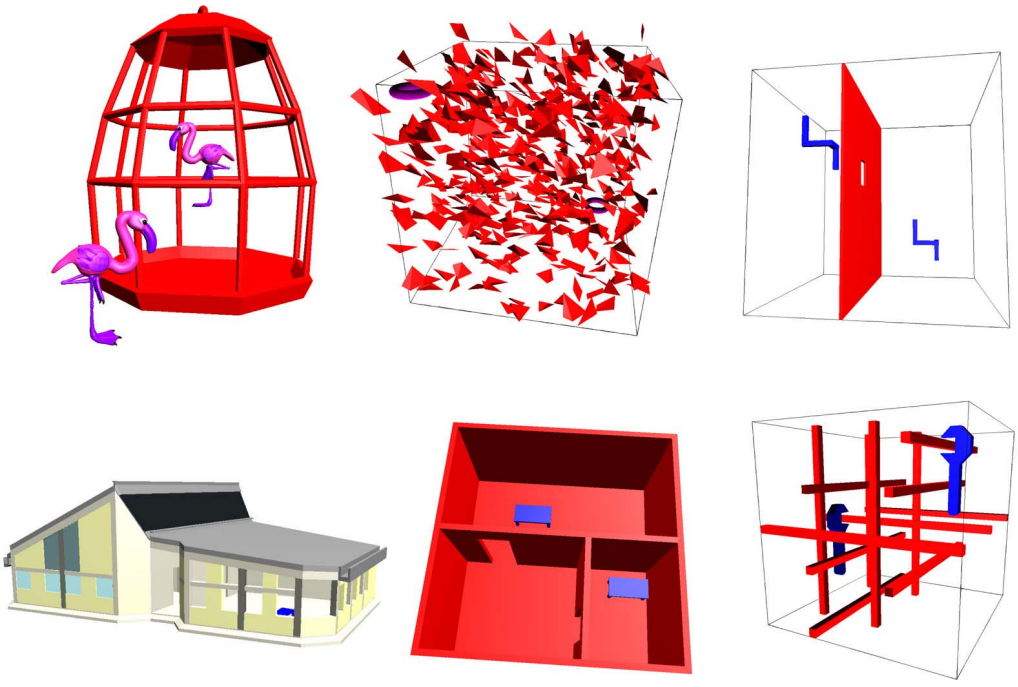


Fig. 2. The six scenes used for testing.

These are the most crucial steps and they strongly influence the running time and the structure of the roadmap graph.

Furthermore, we focus on multiple shot techniques and will not consider single shot methods such as RRT-based planners [23].

3. 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. We will investigate this in future work.

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 the basic collision checking package [37]. 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 Fig. 2). The cage and wrench scenes are borrowed from the Motion Strategy Library [27]. Furthermore, 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 object (table) is small compared to the house. Because the 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 object (156 triangles) in a small workspace (48 triangles). There are many different solutions. At the start and goal the object is rather constrained.

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 [37]. 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 collisions 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) [33].

In early papers on PRM the incremental method was used. Later papers suggest that the binary method works better [34]. 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.

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 [10]).

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 do we perform 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.

rotate-at-s: While the previous methods check collisions along a straight line, the rotate-at-s approach first translates from start to an intermediary configuration s halfway, then rotates, and finally translates to the endpoint [1]. We set s to 0.5.

Table 1 summarizes the results (using deterministic Halton points for sampling and a simple nearest-k node adding strategy; see below).

Table 1
Running times for different collision checking methods

	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

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 [10]. This might be due to the way Solid performs the collision checking with the line segment. In [10] 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.

Also the rotate-at-s technique did not give the improvement suggested in [1]. 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 without line checks.

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

Halton: In [7] 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. [8]). It has been suggested in [7] 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 [41]. The claims in [7] should still hold because they are independent of the seed. By choosing a random seed we avoid the situation in which seed zero is lucky or unlucky.

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. (We set k to 0.2.) So the area becomes smaller when more and more points are added. This

Table 2
Comparison of average running times of six basic sampling strategies

	Basic sampling strategy					
	random	**grid**	**Halton**	**Halton***	**rnd halt.**	**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**

added randomness is another way suggested to 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. A similar approach was employed in [32].

An important question is how to choose random values. Random values were obtained by the Mersenne Twister [28]. Sampling for the rotational degrees of freedom ($SO3$) was performed by choosing random unit quaternions [22]. See [42] for a more extensive elaboration on sampling methods for $SO3$.

Table 2 summarizes the results. It can be seen that the results were rather varying. In general the differences were 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 s or less, while another third took more than 100 s (one run even took 650 s). As a result, conclusions based on average times are difficult to make. Also the other approaches had a high variation in running time. Fig. 3 shows a box plot 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 for example 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.

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.

6. 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.

Table 3
Comparison of average running times of six advanced sampling strategies

	Sampling around obstacles						
	Gaussian	**obstacle**	**obstacle***	**bridge**	**MA**	**NC**	**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**

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 $\mathcal{C}_{\text{forb}}$ we add the free sample. This leads to a favorable sample distribution [5].

obstacle based: This technique, based on [2], 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 is a hybrid technique that aims at better coverage of the free space [15]. 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 $\mathcal{C}_{\text{free}}$ is the free sample added. (To also get points in open space, every sixth sample is simply chosen randomly.)

medial axis: This technique generates samples near the medial axis (MA) of the free space [40]. All samples have two 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 $\mathcal{C}$-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 $\mathcal{C}$-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 3 shows the results. (The Halton* approach is shown for comparison.)

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 ten times slower. The medial axis approach was even worse, due to the expensive calculations. The method does though give samples that are

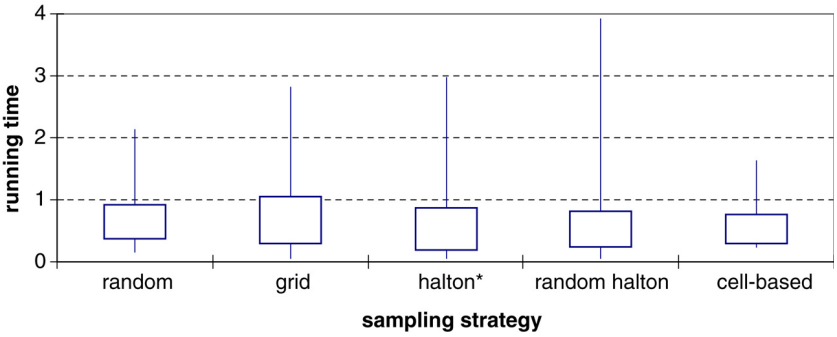


Fig. 3. Box plot for the rooms scene, showing the variation in running time for 30 runs. A vertical line represents the minimum and maximum value of the experiments and the box represents the middle fifty per cent of the data.

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; see e.g. [18].

7. Node adding

In this section we will study 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. An important question here is what we use as a distance measure; see also [1]. In this paper we use $d = d_t + d_r$, where d_t denotes the translational distance of the origin of the object and d_r denotes the rotational distance between two unit quaternions that represent the start and goal orientation; see [22]. This measure is easy to compute and gives a reasonable estimate.

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

Table 4
Comparison between four node adding strategies

	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

We consider the following techniques.

nearest-n: 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: 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: 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 [30]. 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. It has been observed in [30] that the number of useful nodes remains small, making it possible to try connections to all of them.

Table 4 summarizes the results. Although the visibility approach pruned the graph a lot, it still performed worse on most scenes. Only for the hole scene did it perform better. We feel that the reason is that the approach is too strict in

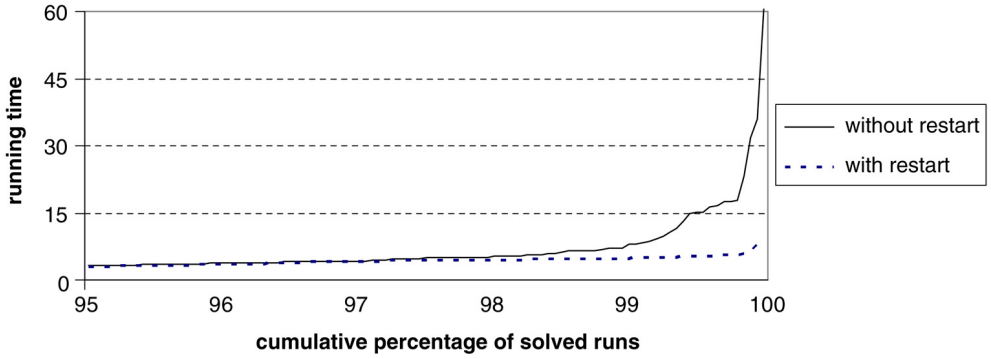


Fig. 4. Cumulative percentages of 2000 runs solved in t seconds for the wrench scene.

rejecting nodes. In general, the component-based techniques were faster except for the hole scene. We must remark that for the hole scene it is better to use an obstacle-based sampling technique such as nearest contact. For this technique, nearest-k was actually three times faster than the visibility approach (and the components techniques).

8. Variation of running times

An advantage of random sampling techniques is that they are usually fast and can successfully 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 for two reasons. First, a large variation complicates statistical analysis and can even make it unreliable. Second, it is undesirable from a user's point of view; for example, 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 [16], where a 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. Fig. 4 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 s). By restarting the PRM after 4 s 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.

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

	Statistics for the variance problem			
	minimum	**average**	**maximum**	**st. dev.**
Without restart	0.03	1.01	60.81	2.32
With restart	0.05	0.88	8.20	1.00

9. Conclusions

In this paper we have 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. The results showed that many claims on efficiency of certain sampling approaches could not be verified, i.e., there was little difference between the various uniform sampling methods and they were only outperformed by advanced sampling methods for special (narrow passage) cases.

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.

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 nearest contact and visibility works best. But experiments showed that this combination is actually about three times worse than the best combination.

The study also showed the difficulty of evaluating the quality of the techniques. In particular the variance in the running time and the influence of certain bad runs were surprisingly large. We presented a very simple variance killer that seems to be effective. We are though fairly 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. Also, further study for other robot types such as articulated and car-like robots would be

interesting since we only compared techniques for free-flying objects.

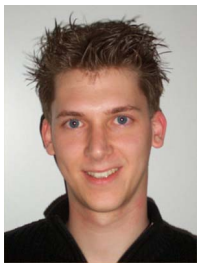
We hope that our studies shed some more light on the questions of what technique to use in which situation. A major challenge is to create planners that automatically choose the correct combination of techniques based on scene properties or that learn the correct settings while running.

Acknowledgements

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 Dennis Nieuwenhuisen who created part of the SAMPLE software.

References

[1] N.M. Amato, O. Bayazit, L. Dale, C. Jones, D. Vallejo, Choosing good distance metrics and local planners for probabilistic roadmap methods, in: IEEE Int. Conf. on Robotics and Automation, 1998, pp. 630–637.
[2] N.M. Amato, O. Bayazit, L. Dale, C. Jones, D. Vallejo, OBPRM: An obstacle-based PRM for 3D workspaces, in: Workshop on the Algorithmic Foundations of Robotics, 1998, pp. 155–168.
[3] N.M. Amato, Y. Wu, A randomized roadmap method for path and manipulation planning, in: IEEE Int. Conf. on Robotics and Automation, 1996, pp. 113–120.
[4] J. Barraquand, L.E. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, P. Raghavan, A random sampling scheme for path planning, International Journal of Robotics Research 16 (1997) 759–774.
[5] R. Bohlin, Motion planning for industrial robots. Ph.D. Thesis, Göteborg University, 1999.
[6] R. Bohlin, L.E. Kavraki, Path planning using lazy PRM, in: IEEE Int. Conf. on Robotics and Automation, 2000, pp. 521–528.
[7] M. Branicky, S. LaValle, K. Olson, L. Yang, Quasi-randomized path planning, in: IEEE Int. Conf. on Robotics and Automation, 2001, pp. 1481–1487.
[8] B. Chazelle, The Discrepancy Method, Cambridge University Press, Cambridge, 2000.
[9] J. Cortes, T. Simeon, J.P. Laumond, A random loop generator for planning the motions of closed kinematic chains using PRM methods, in: IEEE Int. Conf. on Robotics and Automation, 2002, pp. 2141–2146.
[10] L. Dale, Optimization techniques for probabilistic roadmaps. Ph.D. Thesis, Texas A&M University, 2000.
[11] L. Dale, N.M. Amato, Probabilistic roadmaps – putting it all together, in: IEEE Int. Conf. on Robotics and Automation, 2001, pp. 1940–1947.
[12] R. Geraerts, M.H. Overmars, A comparative study of probabilistic roadmap planners, in: Workshop on the Algorithmic Foundations of Robotics, 2002, pp. 43–57.
[13] L. Han, N.M. Amato, A kinematics-based probabilistic roadmap method for closed chain systems, in: Algorithmic and Computational Robotics: New Directions. The Fourth Workshop on the Algorithmic Foundations of Robotics, 2000, pp. 233–245.
[14] C. Holleman, L.E. Kavraki, J. Warren, Planning paths for a flexible surface patch, in: IEEE Int. Conf. on Robotics and Automation, 1998, pp. 21–26.
[15] D. Hsu, T. Jiang, J. Reif, Z. Sun, The bridge test for sampling narrow passages with probabilistic roadmap planners, in: IEEE Int. Conf. on Robotics and Automation, 2003, pp. 4420–4426.
[16] D. Hsu, T. Jiang, J. Reif, Z. Sun, On addressing the run-cost variance in randomized motion planners, in: IEEE Int. Conf. on Robotics and Automation, 2003, pp. 2934–2939.
[17] D. Hsu, L.E. Kavraki, J.-C. Latombe, R. Motwani, S. Sorkin, On finding narrow passages with probabilistic roadmap planners, in: Robotics: The Algorithmic Perspective. The Third Workshop on the Algorithmic Foundations of Robotics, 1998, pp. 141–154.
[18] D. Hsu, Z. Sun, Adaptive hybrid sampling for probabilistic roadmap planning, Technical Report TRA5/04, National University of Singapore, 2004.
[19] L.E. Kavraki, Random networks in configuration space for fast path planning, Ph.D. Thesis, Stanford University, 1995.
[20] L.E. Kavraki, J.-C. Latombe, Randomized preprocessing of configuration space for fast path planning, in: IEEE Int. Conf. on Robotics and Automation, 1994, pp. 2138–2145.
[21] L.E. Kavraki, P. Švestka, J.-C. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Transactions on Robotics and Automation 12 (1996) 566–580.
[22] J.J. Kuffner, Effective sampling and distance metrics for 3D rigid body path planning, in: IEEE Int. Conf. on Robotics and Automation, 2004, pp. 3993–3998.
[23] J.J. Kuffner, S.M. LaValle, RRT-connect: An efficient approach to single-query path planning, in: IEEE Conf. on Robotics and Automation, 2000, pp. 995–1001.
[24] F. Lamiraux, L. Kavraki, Planning paths for elastic objects under manipulation constraints, International Journal of Robotics Research 20 (3) (2001) 188–208.
[25] J.-C. Latombe, Robot Motion Planning, Kluwer, 1991.
[26] J.-C. Latombe, Motion planning: A journey of robots, molecules, digital actors, and other artifacts, International Journal of Robotics Research 18 (1999) 1119–1128.
[27] S.M. LaValle, Planning Algorithms, http://msl.cs.uiuc.edu/planning, 2004.
[28] M. Matsumoto, T. Nishimura, Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator, Transactions on Modelling and Computer Simulation 8 (1998) 3–30.
[29] D. Nieuwenhuisen, M.H. Overmars, Using cycles in probabilistic roadmap graphs, in: IEEE Int. Conf. on Robotics and Automation, 2004, pp. 446–452.
[30] C. Nissoux, T. Siméon, J.-P. Laumond, Visibility based probabilistic roadmaps. in: IEEE Int. Conf. on Intelligent Robots and Systems, 1999, pp. 1316–1321.
[31] M.H. Overmars, A random approach to motion planning, Technical Report RUU-CS-92-32, Utrecht University, 1992.
[32] B. Salomon, M. Garber, M.C. Lin, D. Manocha, Interactive navigation in complex environments using path planning, in: Symposium on Interactive 3D Graphics, 2003, pp. 41–50.
[33] G. Sánchez, J.-C. Latombe, A single-query bi-directional probabilistic roadmap planner with lazy collision checking, in: Int. Symposium on Robotics Research, 2001, pp. 403–418.
[34] G. Sánchez, J.-C. Latombe, On delaying collision checking in PRM planning — application to multi-robot coordination, International Journal of Robotics Research 21 (1) (2002) 5–26.
[35] T. Simeon, J. Cortes, A. Sahbani, J.P. Laumond, A manipulation planner for pick and place operations under continuous grasps and placements, in: IEEE Int. Conf. on Robotics and Automation, 2002, pp. 2022–2027.
[36] P. Švestka, Robot motion planning using probabilistic road maps. Ph.D. Thesis, Utrecht University, 1997.
[37] G. van den Bergen, Collision Detection in Interactive 3D Environments, Morgan Kaufmann, 2003.
[38] P. Švestka, M.H. Overmars, Motion planning for car-like robots, a probabilistic learning approach, International Journal of Robotics Research 16 (1997) 119–143.
[39] P. Švestka, M.H. Overmars, Coordinated path planning for multiple robots, Robotics and Autonomous Systems 23 (1998) 125–152.
[40] S.A. Wilmarth, N.M. Amato, P.F. Stiller, MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space, in: IEEE Int. Conf. on Robotics and Automation, 1999, pp. 1024–1031.
[41] X. Hickernell, F, J. Wang, Randomized Halton sequences, Mathematical and Computer Modelling 32 (2000) 887–899.
[42] A. Yershova, S.M. Lavalle, Deterministic sampling methods for spheres and $SO(3)$, in: IEEE Int. Conf. on Robotics and Automation, 2004, pp. 3974–3980.



Roland Geraerts received his M.S. degree in Computer Science in 2001 from Utrecht University in the Netherlands. Currently he is a Ph.D. candidate at the Department of Computer Science at the same university. His interests include robotics and motion planning.

Mark Overmars received his Ph.D. degree in Computer Science in 1983 from Utrecht University in the Netherlands. Currently he is a full professor at the Department of Computer Science at the same university. Here he heads the Center for Geometry, Imaging and Virtual Environments. His main research interests include computational geometry and its application in areas like virtual reality, robotics, and computer games.