Recent Developments in Motion Planning*

Mark H. Overmars

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

Abstract. Motion planning is becoming an important topic in many application areas, ranging from robotics to virtual environments and games. In this paper I review some recent results in motion planning, concentrating on the probabilistic roadmap approach that has proven to be very successful for many motion planning problems. After a brief description of the approach I indicate how the technique can be applied to various motion planning problems. Next I give a number of global techniques for improving the approach, and finally I describe some recent results on improving the quality of the resulting motions.

1 Introduction

Automated motion planning is rapidly gaining importance in various fields. Originally the problem was mainly studies in robotics. But in the past few years many new applications arise in fields such as animation, computer games, virtual environments, and maintenance planning and training in industrial CAD systems.

In its simplest form the motion planning problem can be formulated as follows: Given a moving object at a particular start position in an environment with a (large) number of obstacles and a required goal position, compute a collision free path for the object to the goal. Such a path should preferably be short, "nice", and feasible for the object.

The motion planning problem is normally formulated in the configuration space of the moving object. This is the space of all possible configurations of the object. For example, for a translating and rotating object in the plane the configuration space is a 3-dimensional space where the dimensions correspond to the x and y coordinate of the object and the rotation angle θ. For a robot arm consisting of n joints, the configuration space is n-dimensional space where each dimension corresponds to a joint position. A motion for the robot can be describes as a curve in the configuration space.

Over the years many different techniques for motion planning have been devised. See the book of Latombe[16] 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 for many more recent results.

Motion planning approaches can globally be subdivided in three classes: cell-decomposition methods, roadmap methods, and potential field (or local) methods. Cell decomposition methods try to divide the free part of the configuration

* This research has been supported by the ESPRIT LTR project MOLOG.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2331, pp. 3–13, 2002.

space (that is, those configurations that do not cause collisions) into a number of cells. Motion is than planned through these cells. Unfortunately, when the dimension of the configuration space gets higher or when the complexity of the scene is large, the number of cells required becomes too large to be practical. Roadmap methods try to construct a network of roads through the configuration space along which the object can move without collision. This roadmap can be seen as a graph, and the problem is reduced to graph searching. Unfortunately, computing an effective roadmap is very difficult. Potential field methods and other local methods steer the object by determining a direction of motion based on local properties of the scene around the moving object. The object tries to move in the direction of the goal while being pushed away by nearby obstacles. Because only local properties are used the object might move in the wrong direction, which can lead to dead-lock situations. Also there are some approaches based on neural networks (e.g. [26]) and genetic algorithms (e.g. [3]).

The *probabilistic roadmap planner (PRM)*, also called the probabilistic path planner (PPP), is a relatively new approach to motion planning, developed independently at different sites [2, 11, 13, 18, 22]. It is a roadmap technique but rather than constructing the roadmap in a deterministic way, a probabilistic technique is used. A big advantage of PRM is that its complexity tends to be dependent on the difficulty of the path, and much less on the global complexity of the scene or the dimension of the configuration space.

In the past few years the method has been successfully applied in many motion planning problems dealing with robot arms[14], car-like robots[23, 25], multiple robots[24], manipulation tasks[21] and even flexible objects[9, 15]. In all these cases the method is very efficient but, due to the probabilistic nature, it is difficult to analyze (see e.g. [12]).

In this paper I will give an overview of the probabilistic roadmap approach and indicate some of the recent achievements. After a brief description of the basic technique in Sect. 2 I will show how the approach can be used for solving various types of motion planning problems. Then, in Sect. 4, I will describe a number of interesting improvements that have been suggested. Finally, in Sect. 5, I will discuss a number of issues related to the quality of the resulting motions.

2 Probabilistic Roadmap Planner

The motion planning problem is normally formulated in terms of the *configuration space* $\mathcal{C}$, the space of all possible configurations of the robot. Each degree of freedom of the robot corresponds to a dimension of the configuration space. Each obstacle in the workspace, in which the robot 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 of the robot 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 $\mathcal{C}_{\text{free}}$.

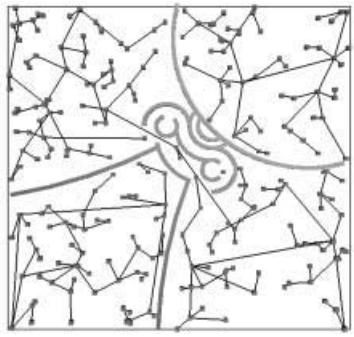

Fig. 1. A typical graph produced by PRM.

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. Here we base ourselves on the description in [22].

The global idea of PRM is to pick a collection of (random) configurations in the free space $\mathcal{C}_{\text{free}}$. These free configurations form the nodes of a graph $G = (V, E)$. A number of 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. (It must also be deterministic.) A typical choice is to use a simple interpolation between the two configurations, and then check whether the path is collision-free. See Fig. 1 for an example of a graph created with PRM in a simple 2-dimensional scene. (Because the configuration space is 3-dimensional, the graph should actually be drawn in this 3-dimensional space. In the figure and in all other figures in this paper we project the graph back into the workspace.)

Once the graph reflects the connectivity of $\mathcal{C}_{\text{free}}$ it can be used to answer motion planning queries. 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 robot. In a post-processing step this path is then smoothed to improve its quality. The pseudo code for the algorithm for constructing the graph is shown in Algorithm ConstructRoadmap.

There are many details to fill in in this global scheme: which local planner to use, how to select promising pairs of nodes to connect, what distance measure to use, how to improve the resulting paths, etc. These typically depend on the type of motion planning problem we want to solve. See Sect. 3 for some information about this.

If we already know the start and goal configuration, we can first add them to the graph and continue the loop until a path between start and goal exists.

Algorithm 1 ConstructRoadmap

Let: $V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
1: **loop**
2: $c \leftarrow$ a (random) configuration in $\mathcal{C}_{\text{free}}$
3: $V \leftarrow V \cup \{c\}$
4: $N_c \leftarrow$ a set of 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

Note that the test in line 6 guarantees that we never connect two nodes that are already connected in the graph. Although such connections are indeed not necessary to solve the problem, they can still be useful for creating shorter paths. See Sect. 5 for details.

The two time-consuming steps in this algorithm are line 2 where a free sample is generated, and line 7 where we test whether the local method can find a path between the new sample and a configuration in the graph. The geometric operations required for these steps dominate the work. So to improve the efficiency of PRM we need to implement these steps very efficiently and we need to avoid calls to them as much as possible. That is, we need to place samples at "useful" places and need to compute only "useful" edges. The problem is that it is not clear how to determine whether a node or edge is "useful". Many of the improvements described in Sect. 4 work this way.

Because of the probabilistic nature of the algorithm it is difficult to analyze it. The algorithm is not complete. It can never report that for certain no solution exists. But fortunately for most applications the algorithm is probabilistically complete, that is, when the running time goes to infinity, the chance that a solution is found goes to 1 (assuming a solution exists). Little is known about the speed of convergence[12]. In practice though solutions tend to be found fast in most cases.

3 Applications

The simplest application of PRM is an object that moves freely (translating and rotating) through a 2- or 3-dimensional workspace. In this case the configuration spaces is either 3-dimensional or 6-dimensional. As a local planner we can use a straight-line interpolation between the two configuration. (An interesting question here is how to represent the rotational degrees of freedom and how to interpolate between them but we won't go into detail here.) As distance between two configurations we must use a weighted sum of the translational distance and the amount of rotation. Typically, the rotation becomes more important when the moving object is large. With these details filled in the PRM approach can be applied without much difficulty. (See though the remarks in the next sections.) For other types of moving objects there is some more work to be done.

Car-like Robots A car-like robot has special so-called non-holonomic constraints than restrict its motion. For example, a car cannot move sideways. Still the configuration space is 3-dimensional because, given enough space, the car can get in each position in any orientation. Using a simple straight-line interpolation for the local planner no longer leads to valid paths for the robot. So we need to use a different local planner. One choice, used e.g. in [22, 23], is to let the local planner compute paths consisting of a circle arc of minimal turning radius, followed by a straight-line motion, followed by another circle arc. It was shown in [23] that such a local planner is powerful enough to solve the problem. The approach is probabilistically complete. Extensions have also been proposed towards other types of robots with non-holonomic constraints, like trucks with trailers[25].

Robot Arms A robot arm has a number of degrees of freedom depending on the number of joints. Typical robot arms have up to 6 joints, resulting in a 6-dimensional configuration space. Most of these are rotational degrees of freedom, often with limits on the angle. The PRM approach can be applied rather easily in this situation. As local method we can interpolate between the configurations (although there exist better methods, see [14]). When computing distances it is best to let the major axis of the robot play a larger role than the minor axis. Again the approach is probabilistically complete.

Multiple Robots When there are multiple moving robots or objects in the same environment we need to coordinate their motions. There are two basic approaches for this (see e.g. the book of Latombe[16]). When applying centralized planning the robots together are considered as one robotic system with many degrees of freedom. For example in the situation in Fig. 2 there are 6 robot arms with a total of 36 degrees of freedom. When applying decoupled planning we first compute the individual motions of the robots and then try to coordinate these over time. This is faster but can lead to deadlock. In [24] a solution based on PRM is proposed that lies between these two. Rather that coordinate the paths, the roadmaps themselves are coordinated, leading to a faster and probabilistically complete planner.

In a recent paper Sánchez and Latombe[19] show that with a number of improvements the PRM approach can be successfully applied to solve complicated motion planning with up to 6 robot arms, as shown in Fig. 2. When the number of robots is much larger the problem though remains unsolved.

Other Applications The PRM approach has been successfully applied in many other situations. Applications include motion planning for flexible objects[9, 15], motion planning with closed kinematic loops[7, 6], like two mobile robot arms that together hold an object, motion planning in the presence of dangerzones that preferably should be avoided[20], and manipulation tasks[21]. In all these cases one need to find the right representation of the degrees of freedom of the problem, construct an appropriate local planner and fill in the parameters of the PRM approach. It shows the versatility of the method.

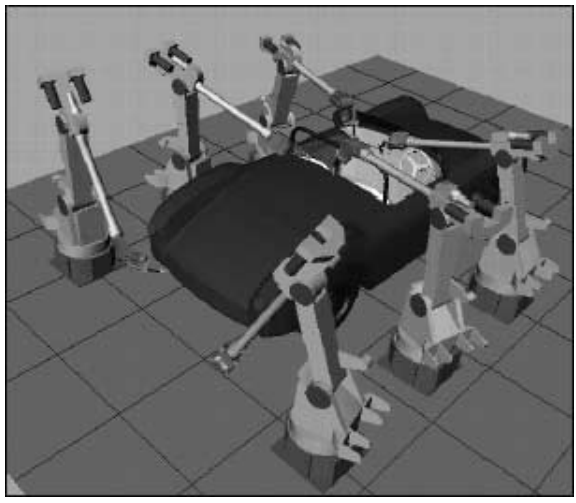

Fig. 2. An example where 6 robots must plan their motions together (taken from Sánchez and Latombe[19]).

4 Improving the Approach

Although the PRM approach can solve many different types of motion planning problems effectively there are a number of problematic issues. Here we discuss some improvements that have been proposed.

Sampling Strategy The default sampling approach samples the free space in a uniform way. This is fine when obstacle density is rather uniform over the scene but in practice this assumption is not correct. Some areas tend to be wide open while at other places there are narrow passages (in particular in configuration space). To obtain enough random samples in such narrow passages one would need way too many samples in total. So a number of authors have suggested ways to obtain more samples in difficult areas.

One of the early papers[14] suggested to maintain information on how often the local planner fails for certain nodes in the graph. When this number is large for a particular node this suggest that this node is located in a difficult area. The same is true when two nodes lie near to each other but no connection has been found between them. One can increase the number of samples in such areas.

Another approach is to place addition samples near to edges and vertices of obstacles[1, 23] or to allow for samples inside obstacles and pushing them to the outside[27, 10]. Such methods though require more complicated geometric operations on the obstacles.

An approach that avoids such geometric computations is the Gaussian sampling technique[5]. The approach works as follows. Rather than one sample we take two samples where the distance between the two samples is taken with respect to a Gaussian distribution. When both samples are forbidden we obviously

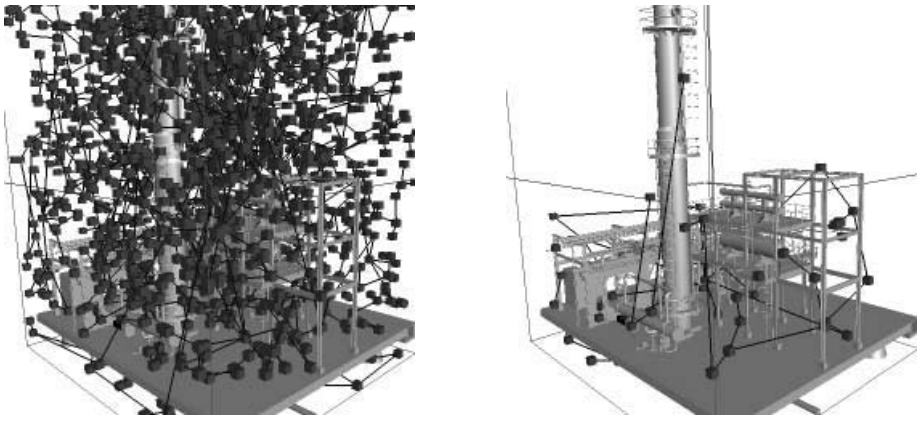

Fig. 3. A motion planning problem in a complex industrial environment with over 4000 obstacles. The left picture shows the nodes obtained with uniform sampling, and the right picture the nodes obtained with Gaussian sampling.

remove them. When both lie in the free space we also remove them because there is a high probability that they lie in an open area. When only one of the two samples is free we add this sample to the graph. It can be shown (see [5]) that this approach results in a sample distribution that corresponds to a Gaussian blur of the obstacles (in configuration space). The closer you are to an obstacle, the higher the change that a sample is placed there. See Fig. 3 for an example.

Roadmap Size As indicated above, computing paths using the local planner is the most time-consuming step in the PRM algorithm. We would like to avoid such computations as much as possible. One way to do this is to keep the roadmap as small as possible. The visibility based PRM[17] only adds a node to the roadmap if it either can be connected to two components of the graph or to no component at all. The reason is that a node that can be connected to just one component represents an area that can already be "seen" by the roadmap. It can be shown that the approach converges to a roadmap that covers the entire free space. The number of nodes tends to remain very small, unless the free space has a very complicated structure.

Another idea is not to test whether the paths are collision free unless they are really needed[4]. Such a lazy approach only checks whether the nodes are collision free and when nodes are close to each other they are connected with an edge. Only when an actual motion planning query must be solved we test whether the edges on the shortest path in the graph are collision-free. If not we try other edges, until a path is found. The rational behind this is that for most paths we only need to consider a small part of the graph before a solution is found. In [19] a similar idea is used. Here it is also argued and demonstrated that the chance that an edge is collision-free is large when the endpoints (the nodes) are collision-free and the length of the edge is short.

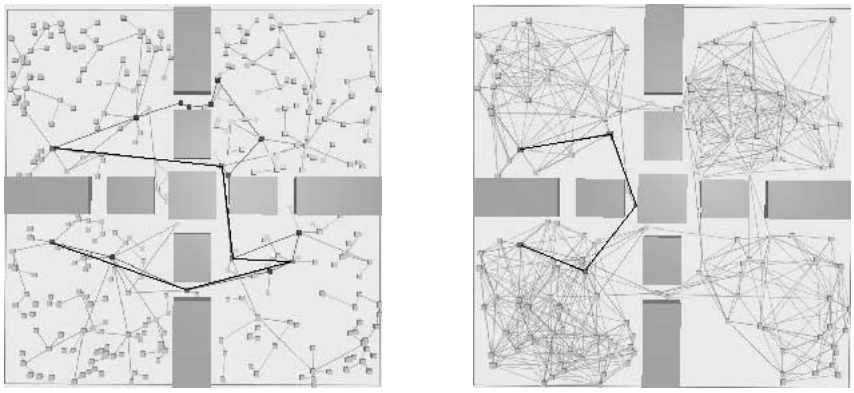

Fig. 4. The left picture shows the graph in the default algorithm. Here a long detour is made. In the right picture cycles are added and the length of the path is reduced considerably.

5 Path Quality

One of the problems of the PRM approach is that the resulting motions are ugly. This is due to the random nature of the samples. A resulting path can make long detours and contain many redundant motions. Also the path normally consists of straight-line motions (in the configuration space) leading to first-order discontinuities at the nodes of the graph. In most applications such ugly paths are unacceptable.

The standard method used to remedy these problems is to smooth the resulting path in a post-processing phase. This smoothing technique consists of taking random pairs (c_1, c_2) of configurations on the path (not necessarily nodes of the graph) and trying to replace the path between c_1 and c_2 by the path resulting from calling the local planner on (c_1, c_2), if this new path is collision free. Unfortunately, smoothing only partially solves the problem. It does reduce the length of the path in open areas but it often cannot correct long detours around obstacles. Also it does not make the path first-order continuous and the path can still include many redundant (rotational) motions, in particular in a 3-dimensional workspace.

In this section we will discuss some recent approaches to improving the path quality. More details will be given in an upcoming paper[8].

Length A prime reason why paths computed with PRM are too long is that a tree (or to be more precise, a forest) is used as roadmap. The advantage of this is that it will save computation time, because less calls to the local planner are required, while connectivity is maintained. So the obvious solution is to add additional edges, leading to cycles in the roadmap. This is easier said than done because we want to avoid calls to the local planner as much as possible (because this is the most time consuming operation in the algorithm). So we only want

to create a cycle when it is "useful". We define useful as follows: Assume the algorithm is trying to add configuration c to the graph. Let c' be a node in the neighbor set N_c. We try to add an edge between c and c' when they are not yet connected in the graph or when the current distance d_G of the shortest path in the graph is larger than $k.d(c, c')$ for some given constant parameter k. So we only try to add the edge when it would improve the length of the shortest path with a factor at least k. The parameter k will determine how dense the graph will be. See Fig. 4 for an example.

There is one algorithmic problem left. To implement the approach we need to be able to compute a shortest path in the graph whenever we try to add an edge. This is rather expensive and would dominate the cost of the algorithm when the graph gets large (it will be called a quadratic number of times). The solution is based on the observation that we can stop searching the graph when shortest paths in the graph become longer than $k.d(c, c')$. We can than immediately decide to add the edge. This will prune the graph quite a bit. We can take this one step further by also taking the distance between the current node in the graph search and c' into account. This leads to some sort of A* algorithm that is a lot faster.

Smoothness Nodes in the graph introduce first-order discontinuities in the motion. We would like to avoid this. This can be achieved as follows. Let e and e' be two consecutive edges in the final path. Let p_m be the midpoint of e and p'_m be the midpoint of e'. We replace the part of the path between p_m and p'_m by a circle arc. This arc will have its center on the bisecting line of e and e', will touch e and e' and have either p_m or p'_m on its boundary. Doing this for each consecutive pair of edges results in a smooth path. The only problem is that the motion along the circle arc might collide with an obstacle. In this case we make the circle smaller, pushing it more towards the node between the edges. It is easy to verify that there always exists a circle arc between the edges that does not introduce collisions. Hence, the method is complete. See Fig. 5 for an example.

When the angle between two consecutive edges becomes small, the radius of the circle becomes small as well. We often like to avoid this. We are currently investigating how we can produce roadmaps that keep the angles as large as possible.

Redundant Motions Allowing cycles in graphs and smoothing the path improves the motion a lot. Still redundant motions can occur. For example, the object can continuously spin around its center. Such motion does not really increase the time it takes to execute the motion. Hence standard smoothing techniques tend not to work. One could add a penalty factor in the length of the path but this again does often not help.

There are a number of techniques that try to remedy this problem. One is to add many nodes with the same orientation. (Or stated in a more generic way, divide the degrees of freedom in major degrees of freedom and minor ones and generate many configurations with the same values for the minor degrees of

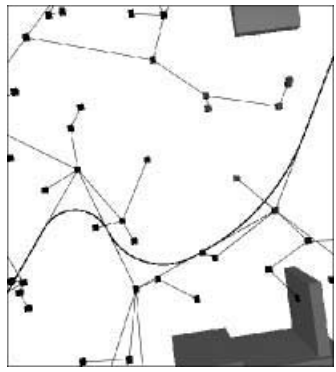

Fig. 5. An example of a part of a path with circular blends.

freedom.) A similar idea was used in a paper by Lamiraux and Kavraki on moving flexible objects[15]. A second approach is to do the smoothing in a different way. The standard smoothing technique replaces pieces of the path by calls to the local planner, that is, by a straight line in the configuration space. In this way all degrees of freedom are smoothed at the same moment. But some of them might be necessary while others are not. For example, the translational degrees of freedom might be necessary to get the object around an obstacle while the rotational degrees of freedom are not necessary. By smoothing the degrees of freedom one at a time we create better paths. Finally, we can try to find a better path by resampling the configuration space in a tube around the original path, similar to the technique in [25].

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, Y. Wu, A randomized roadmap method for path and manipulation planning, *Proc. IEEE Int. Conf. on Robotics and Automation,* 1996, pp. 113–120.
3. P. Bessière. J.M. Ahuactzin, E.-G. Talbi, E. Mazer, The Ariadne's clew algorithm: Global planning with local methods, in: K. Goldberg et al. (eds.), *Algorithmic foundations of robotics*, A.K. Peters, 1995, pp. 39–47.
4. R. Bohlin, L.E. Kavraki, Path planning using lazy PRM, *Proc. IEEE Int. Conf. on Robotics and Automation,* 2000, pp. 521–528.
5. 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.
6. J. Cortes, T. Simeon, J.P. Laumond, A random loop generator for planning the motions of closed kinematic chains using PRM methods, Rapport LAAS N01432, 2001.
7. 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.

8. O. Hofstra, D. Nieuwenhuisen, M.H. Overmars, Improving path quality for probabilistic roadmap planners, 2002, in preparation.
9. 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.
10. 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.
11. L. Kavraki, *Random networks in configuration space for fast path planning,* PhD thesis, Stanford University, 1995.
12. L. Kavraki, M. Kolountzakis, J.C. Latombe, Analysis of probabilistic roadmaps for path planning, *Proc. IEEE Int. Conf. on Robotics and Automation,* 1996, pp. 3020–3025.
13. 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.
14. 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.
15. F. Lamiraux, L.E. Kavraki, Planning paths for elastic objects under manipulation constraints, *Int. Journal of Robotics Research* **20** (2001), pp. 188–208.
16. J-C. Latombe, *Robot motion planning*, Kluwer Academic Publishers, Boston, 1991.
17. C. Nissoux, T. Siméon, J.-P. Laumond, Visibility based probabilistic roadmaps, *Proc. IEEE Int. Conf. on Intelligent Robots and Systems,* 1999, pp. 1316–1321.
18. 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.
19. 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.
20. D. Sent, M.H. Overmars, Motion planning in an environment with dangerzones, *Proc. IEEE Int. Conf. on Robotics and Automation,* 2001, pp. 1488–1493.
21. T. Simeon, J. Cortes, A. Sahbani, J.P. Laumond, A manipulation planner for pick and place operations under continuous grasps and placements, Rapport LAAS N01433, 2001.
22. P. Švestka, *Robot motion planning using probabilistic roadmaps,* PhD thesis, Utrecht Univ. 1997.
23. 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.
24. P. Švestka, M.H. Overmars, Coordinated path planning for multiple robots, *Robotics and Autonomous Systems* **23** (1998), pp. 125–152.
25. 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.
26. J. Vleugels, J. Kok, M.H. Overmars, Motion planning with complete knowledge using a colored SOM, *Int. Journal of Neural Systems* **8** (1997), pp. 613–628.
27. 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.