Creating Small Roadmaps for Solving Motion Planning Problems*

Roland Geraerts and Mark H. Overmars
Institute of Information and Computing Sciences
Utrecht University, 3508 TB Utrecht, the Netherlands
{*roland,markov*}*@cs.uu.nl*

***Abstract*— In robot motion planning, many algorithms have been proposed that create a roadmap from which a path for a moving object can be extracted. These algorithms generally do not give guarantees on the quality of the roadmap, i.e. they do not promise that a path will always be found in the roadmap if one exists in the world. Furthermore, such roadmaps often become very large which can cause memory problems and high query times.**

We present a new efficient algorithm that creates small roadmaps for two- and three-dimensional problems. The algorithm ensures that a path is always found (if one exists) at a given resolution. These claims are verified on a broad range of environments. The results also give insight in the structure of covering roadmaps.

***Keywords*—motion planning, small roadmaps, PRM**

I. Introduction

A central problem in robotics is planning a collision-free path for a moving object in a rigid environment. In the last two decades, efficient algorithms have been devised to tackle this problem. They are successfully applied in fields such as mobile robots, manipulation planning, CAD systems, virtual environments, protein folding and human robot planning. See e.g. the books of Latombe [1] and Lavalle [2] for many interesting results.

In these application areas, it is often desirable that a path between a start and goal configuration (which is a placement of the moving object) is quickly found. For example, maintenance studies in industrial installations [3] can be performed more efficiently if an engineer does not have to wait long for a solution. In other fields, e.g. in games, only a fixed small amount of calculation time is available to compute the path. If this calculation takes too long, the game may halt.

Single shot methods, such as RRT [4], aim at quickly connecting a start configuration to a goal configuration. Although good performance is achieved (compared to the traditional methods described in [1]), they may still be too slow as the calculations are performed on-line. In contrast, the Probabilistic Roadmap Method (PRM) enables the construction of a path in real-time [5]–[8]. The PRM consists of two phases: a construction phase (off-line) and a query phase (on-line). In the construction phase, a roadmap (graph) is built, representing the motions that can be made in the environment. Nodes in the roadmap correspond to randomly chosen collision-free configurations. When a node is added, a simple local planner is employed to connect the configuration to some useful neighbors in the roadmap. A neighbor is useful if its distance to the new configuration is less than a predetermined constant. Configurations and connections are added to the roadmap until the roadmap is dense enough. In the query phase, the start and goal configurations are connected to the roadmap. Then, the path can be obtained by a Dijkstra's shortest path query.

A drawback of the PRM is that a resulting roadmap often contains many redundant nodes and edges, in particular when the environment contains one or more narrow passages. Over the years, many improvements have been suggested to tackle the narrow passage problem [9]–[11] but those solutions often lead to large roadmaps. Such large roadmaps increase the time needed to extract a path and can require a vast amount of memory which may not always be available. Furthermore, the roadmap may contain many short edges which complicates the smoothing phase that often follows a query phase [8]. Another drawback is that a path may not always be found in practice while one exists. This occurs for example when the roadmap is not dense enough.

A variant of the PRM that aims at keeping the roadmap small is the visibility based roadmap [12]. This technique only connects configurations to useful nodes. Usefulness is determined as follows: when a new node cannot be connected to 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. Although this approach prunes the roadmap a lot, it is often slower than other variants of the PRM as the approach is too strict in rejecting nodes [6].

We propose a new efficient method that creates a small roadmap for two- and three-dimensional problems. Our method will ensure that a valid query can always be connected to the roadmap and that a path is always found (if one exists) at a given resolution. Because the roadmap is small, smoothing can easily be applied in the preprocessing phase, leading to instant query answers. We will elaborate on the properties of the approach in section II. In section III we show the outline of the method and we work out its details in section IV. In section V we show some experiments on different

*This research was supported by the Dutch Organization for Scientific Research (N.W.O.). This research was also supported 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).

scenes and in section VI we conclude that our algorithm successfully creates small roadmaps for a large diversity of environments.

II. Reachability

An important question is how to determine when the roadmap is dense enough, i.e. when should the algorithm terminate? Many motion planning techniques such as the PRM and RRT are probabilistically complete [5], i.e. whenever a path exists, the probability that it will be found converges to one as the computation time goes to infinity. As there is no guaranteed upper bound, the running time is not a practical termination criterion. Many authors terminate the method when a path between a specified start and goal is found. However, this does not guarantee that this roadmap can solve every query.

In [13], we used the *reachability* criterion to determine when a problem has been solved, i.e. a problem has been solved if a roadmap $G = (V, E)$ covers the free configuration space ($\mathcal{C}_{\text{free}}$) and captures its connectivity. We define coverage and maximal connectivity as follows:

Definition 1 (coverage)**.** *G covers $\mathcal{C}_{\text{free}}$ when each configuration $c \in \mathcal{C}_{\text{free}}$ can be connected using the local planner to at least one vertex $v \in V$.*

Definition 2 (maximal connectivity)**.** *G is maximally connected when for all vertices $v', v'' \in V$, if there exists a path in $\mathcal{C}_{\text{free}}$ between v' and v'', then there exists a path in G between v' and v''.*

Coverage ensures that every query (which consists of a start and goal configuration) can be directly connected to the roadmap, as is required to solve the problem. If there exists a path (in $\mathcal{C}_{\text{free}}$) between the start and goal configuration, then maximal connectivity ensures that a path between them can be found in the roadmap. We will use this criterion as a termination criterion in the remainder of this paper.

Figure 1 shows an environment whose free space is covered by two (white) vertices and is connected via one extra (black) vertex. The reachability region for the upper left vertex has been drawn. Each configuration in this region can be connected with a straight line to the vertex, so a three-node graph suffices to solve this problem. In section IV we will explain how such a reachability region can be computed.

III. Reachability Roadmap

In this section we show how to create a small roadmap that satisfies both the coverage and maximal connectivity criteria. The idea is to compute a small number of *guards* that 'see' the complete free space (coverage). These guards are then connected via *connectors* to fulfill the maximal connectivity criterion. The resulting roadmap is then pruned to obtain an even smaller roadmap.

Computing the minimum number of guards corresponds to the well known art gallery problem which is NP-hard [14]. As

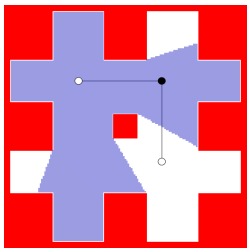


Fig. 1. The coverage and maximal connectivity criteria have been met: the reachability regions of the white vertices cover the free space and are connected via the black vertex.

we want to create a roadmap for a possibly large environment with many obstacles, exact algorithms are infeasible. Therefore, we will use a heuristic to create and place these guards (which are added as nodes to the roadmap).

Globally speaking, the method works as follows. To get the free space covered as fast as possible, we place the guards on locations where they may see a large part of the space. Locations on the medial axis are good candidates as they have a large clearance from the obstacles and thus have an increased probability of having a large reachability region. To prune the number of candidate guards, a new guard is only accepted if it cannot be seen by guards that have already been placed. It can occur that all points on the medial axis are seen by the guards while the free space has not been fully covered yet. In such a case we choose a point in a location which initially cannot be seen by other guards. Then we retract this point to the medial axis to increase the expected size of its reachability region. (Note that this guard will be seen by other guards.) An advantage of placing all guards on the medial axis is that 'good' roadmaps are produced [11]. We keep adding new guards until the free space has been completely covered.

We connect the guards by placing connectors in the overlapping reachability regions of the guards. We consider all pairs of guards whose regions overlap. For each pair, we add the connections and connector to the roadmap.

Finally, we prune this roadmap by transforming it to a Steiner Tree. Such a tree is a shortest network that does not throw away guards (but is allowed to remove connectors) and keeps the connected components connected. Then we add all remaining connections and create a minimal spanning tree to reduce the length.

Theorem 1 (Coverage of $\mathcal{C}_{\text{free}}$)**.** *The above methods lead to a complete coverage of $\mathcal{C}_{\text{free}}$.*

Proof: The medial axis is a connected structure if the free space in which a robot operates is also connected [15]. Hence, the medial axis is a complete representation for motion planning purposes. We start with selecting points on the medial axis and add them as nodes to the roadmap. If all points on the medial axis are covered by the guards but there are still points in $\mathcal{C}_{\text{free}}$ that have not been covered yet, we select such a point which we retract to the medial axis. As

this retraction can be performed in a straight line [11], the original point will be covered by the retracted point. Because we keep adding new guards until the free space is completely covered, the coverage criterion will be met. This criterion will remain satisfied when the roadmap is pruned as these heuristics never remove guards. ■

Theorem 2 (Maximal connectivity of $\mathcal{C}_{\text{free}}$)**.** *The above method satisfies the maximal connectivity criterion.*

Proof: Consider a path Π from node v' to node v'' that lies in $\mathcal{C}_{\text{free}}$. As $\mathcal{C}_{\text{free}}$ is completely covered, there exists for each configuration $\pi \in \Pi$ a node $v_i \in V$ that sees π. Now consider the sequence of different nodes $v_i \in V$ that sees the configurations on Π. (If multiple nodes in V see a particular configuration π, we consider the node v_i having the smallest index i.) If adjacent configurations π_j and π_{j+1} are seen by node v_i, v_i represents the path between these configurations. If they are seen by different nodes v_i and v_{i+1}, there exists a path between these two nodes (possibly via a connector) in the graph as these configurations lie in the overlapping regions of v_i and v_{i+1}. (A degenerate case is when the regions of v_i and v_{i+1} touch and do not overlap. This case can be handled by placing an extra connector on the boundary of these two regions.) Because a path between v' and v'' (in the graph) can be mapped to a path in $\mathcal{C}_{\text{free}}$, this roadmap will obey the maximal connectivity criterion. The maximal connectivity criterion will remain satisfied when the roadmap is pruned as the heuristics only remove edges that are part of cycles. ■

IV. Algorithmic Details

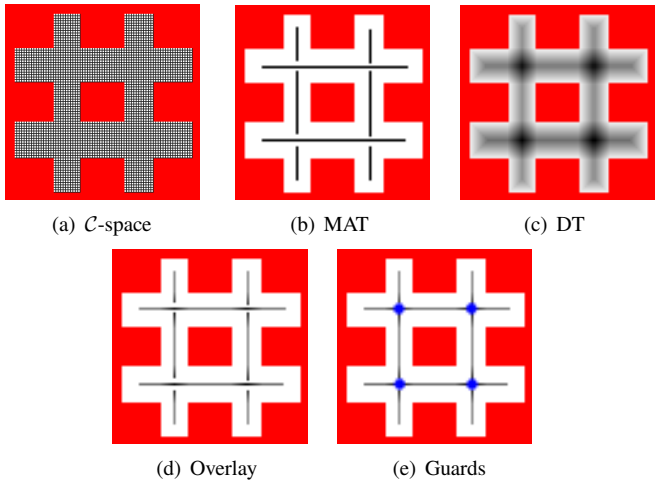
The main operation in our technique is the computation of the reachability regions. An exact computation of these regions would involve intricate and practically infeasible calculations in the configuration space $\mathcal{C}$. To make the calculations feasible, we approximate the $\mathcal{C}$-space by discretizing it. As a consequence, this approach is limited to low-dimensional $\mathcal{C}$-spaces. We performed experiments with both two- and three dimensional $\mathcal{C}$-spaces.

We discretize a d-dimensional $\mathcal{C}$-space as follows. First, we create a d-dimensional grid. For each cell in this grid we check whether the corresponding configuration collides with the obstacles and update the cells appropriately, i.e. we put the value 0 in the cell if it collides and 1 otherwise. We will use this grid in the remainder of the algorithm, see Figure 2(a).

In section IV-A, we show for a given resolution how to get the free space covered and in section IV-B we show how to get this space maximally connected. Finally, we show in section IV-C how the roadmap can be pruned.

A. Coverage

We need a set of cells that reside on the medial axis. This set can be efficiently computed by the medial axis transform



(a) $\mathcal{C}$-space (b) MAT (c) DT

(d) Overlay (e) Guards

Fig. 2. The creation of candidate guards. Figure (a) shows the free (white) cells of the discretized $\mathcal{C}$-space of Figure 1. Figure (b) shows the cells of medial axis transform and (c) the distance transform. Dark cells correspond to a large distance to the closest obstacle while light cells correspond to a small distance. In (d), the overlay of the MAT and DT is shown. Finally, (e) shows the four guards having the largest distance in the overlay.

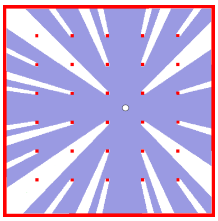
(MAT) which is a well known technique in mathematical morphology. We use the algorithm presented in [16]. This algorithm is a two-pass dynamic program that computes the MAT in O(n) time where n is the number of cells in the grid. See Figure 2(b) for an example.

As guards having a larger distance to the obstacles should be handled earlier than guards having a small distance, we need to compute their distances to the obstacles. These can be efficiently computed by the distance transform (DT) in O(n) time, see Figure 2(c). Like the MAT, the DT is also computed by a two-pass dynamic program, see e.g. [16]. We determine the clearance for each medial axis cell by overlaying the MAT with the DT, see Figure 2(d). These cells are then sorted by decreasing distance using bucket sort in linear time. We store these cells, which are candidates for guards, in a list M, see Figure 2(e).

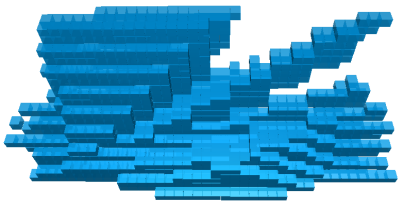
We keep adding guards from this list to the roadmap until the free space has been fully covered (or when all points of M have been handled). As we have already mentioned, we first only add guards that cannot be seen by guards that have already been added. If the space is not covered when all medial axis points have been handled, we consider uncovered points from the distance transform (in decreasing order) and add their retraction on the medial axis, see e.g. [11].

There are now two problems we have to solve, i.e. how to determine the cells that are covered by a guard and how to determine which cells have not been covered yet.

We determine the cells that are covered by a guard placed in cell $m_i \in M$ as follows. For each free cell we check if a straight-line connection between the free cell and cell m_i can be made. By using the Bresenham line-drawing algorithm [17], this line can be computed in linear time in the number



(a) A star-shaped 2D reachability region



(b) A 3D reachability region for the manipulator arm depicted in Figure 5(c)

Fig. 3. Complicated two- and three dimensional discretized reachability regions

of cells covered by the line. Let $|m_i|$ be the number of cells reachable from region m_i and s be the number of cells along the largest side of the grid. Then covering a region using this approach takes $O(|m_i|*s)$ time. However, using a more clever approach we can improve this to $O(|m_i|)$ time. (Details will be given in the full paper.) The shape of such a region can be complicated, see Figure 3.

To determine which cells have not been covered yet, we store in each cell of the grid the set of guards that cover the cell, i.e. when guard m_i is added to the roadmap, for all cells that are covered by this guard index i is inserted to the corresponding sets in the grid. A set is a data structure that allows insertions in O($\log k$) time [18], where k is the number of guards that cover this cell. Checking whether the cell has been covered by a guard is performed in O($\log k$) time. (Storing all covering guards might seem an overkill but we need this information in the next phase.)

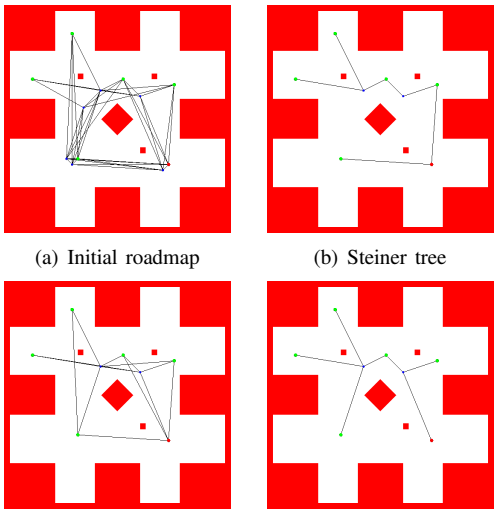
B. Maximal connectivity

As a guard cannot usually see any other guards, we need to calculate a set of connectors to which the guards can be connected such that the maximal connectivity criterion is satisfied. These connectors will be placed in the overlapping reachability regions of the guards. For each pair of guards that share cells in the grid, we place one connector in a cell that lies on the medial axis. (If more than one cell satisfies this condition, we choose the one that has the largest value in the DT.) If such a cell does not exist, we choose a cell that has the largest distance in the DT. (If more than one cell satisfies this condition, we choose the one that minimizes the connection distance of the connector to the two guards.) It is possible that different connectors share the same cell in the grid. In such a case these connectors are merged. This part of the algorithm can be computed in linear time in the number of elements of the sets in the grid. We obtain a roadmap that satisfies the maximal connectivity criterion by adding all collision-free connections between the guards and connectors.

C. Roadmap pruning

If we allow all collision-free connections between guards and connectors, the number of connections can become quadratic in the number of nodes in the roadmap. Figure 4(a) shows such a fully connected roadmap. As our objective is to create a small roadmap, we want to solve the following problem. Given a roadmap G that consists of a set of guards, a set of connectors, and all feasible connections between the joint set of guards and connectors, create a shortest roadmap G' (in terms of total edge length) that spans the guards, see Figure 4(a). This problem is known as the discrete Euclidean Steiner problem which is NP-complete [19]. Consequently, no polynomial time algorithm for this problem is likely to exist. We handle this problem by using the *shortest path heuristic* [19], which is based on Prim's minimum spanning tree algorithm [18]. See Figure 4(b) for an example.

The total edge length of G' can be further decreased by the following approach. First, all collision-free connections between the nodes in G' are added to G', see Figure 4(c). Then, its minimal spanning tree (MST) is calculated by Kruskal's algorithm [18], see Figure 4(d).



(a) Initial roadmap (b) Steiner tree

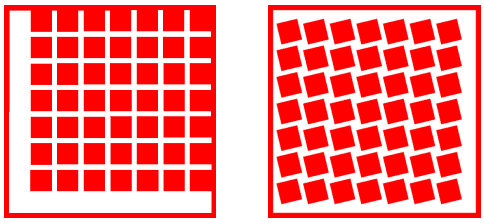
(c) All connections (d) MST

Fig. 4. Roadmap pruning. Figure (a) shows (among other roadmap elements) the connectors (small spheres) that lie in the overlapping regions. From this roadmap, an approximated shortest roadmap is calculated in (b). In (c), all collision-free connections are added. Finally, (d) shows the minimal spanning tree (MST) of the roadmap in (c).

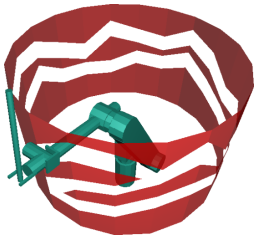
V. Experimental Results

The techniques were implemented in our motion planning system SAMPLE (System for Advanced Motion PLanning Experiments) in Visual C++ under Windows XP. All experiments were run on a 2.66GHz Pentium 4 processor with 1 GB internal memory. SAMPLE used Solid as basic collision checking package [20].

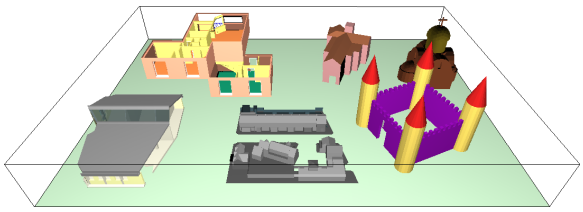
We performed experiments on ten different environments. Due to space limitations, we selected four typical environments for this paper, see Figure 5. (See our website for the results of the remaining experiments [21].) They have the following properties:



(a) Grid (b) Rotated grid



(c) Manipulator



(d) Toy village

Fig. 5. The four test environments

Grid This is a simple 2D environment that contains 49 squares. The optimal solution is a roadmap containing seven guards, six connectors and twelve edges. We want to know whether our algorithm can be competitive compared to the optimal solution.

Rotated grid We rotated the 49 squares which created many narrow passages. We use this environment to study the effect of the reduced local visibility on the size of the roadmaps.

Manipulator This 3D environment features a robot arm with three rotational degrees of freedom that can move through a long passage. We selected this environment to show that our algorithm can handle complex reachability regions in $\mathcal{C}$-space, see Figure 3(b) for an example of such a region.

Toy village This large 3D environment contains seven buildings (>10,000 objects). In this environment there are many scale differences, i.e. the environment has large open spaces (outside) and small rooms (inside). We want to know whether our algorithm can handle such large environments.

For each environment we created a roadmap with our Reachability Roadmap (RR) technique. We compared this technique with the PRM [7] (as this is a widely used motion planning method) and the visibility based roadmap [12] (as this method creates small roadmaps). For the latter two techniques we took the average of 50 independent runs. Such a run was solved if both the coverage and maximal connectivity criteria were satisfied. When we report the construction time for these two methods, we did not include the time needed to check these two criteria.

We recorded the following statistical data: the construction time of the roadmap, the number of nodes $|N|$ and the number of edges $|E|$ in the roadmap. Furthermore, for each environment and technique we measured the length of the roadmap $|G|$ which we calculated as the sum of the length of each edge in the roadmap.

The results are stated in Table I and visualized in Figure 6.

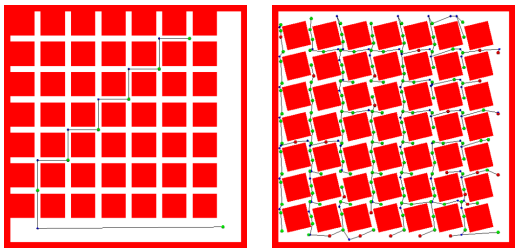
Grid The Reachability Roadmap (RR) technique computed a roadmap that has the minimum number of nodes. The roadmap length is close to optimal. Hence, our algorithm is competitive compared to the optimal solution. While the PRM created much larger roadmaps, the visibility PRM produced reasonable small ones but this took considerably more time than the other two techniques. This is because the visibility PRM sometimes creates artificial narrow passages which complicates solving the problem [12].

Rotated grid As the reachability regions are much smaller in this environment than the regions in the Grid environment, more guards are needed to get the free space covered. The RR technique needed far less nodes and edges than PRM because the RR technique tries to minimize the amount of double coverage. Also, the resulting roadmap length was much smaller. The visibility PRM was not able to solve the problem within one hour.

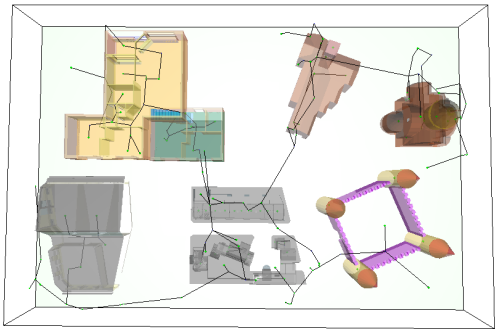
Manipulator The RR technique needed 313 nodes and 306 edges to cover and connect the three-dimensional configuration space. Again, the PRM created much larger roadmaps. This is because the $\mathcal{C}$-space contains narrow passages which complicates connecting the nodes. The visibility PRM was not able to solve the problem within one hour.

Toy village As this environment has a rather non-uniform distribution of the obstacles, the PRM had difficulties covering and connecting the small areas. Also the visibility PRM was not able to solve the problem within one hour. In contrast, the RR technique solved the problem within 10 minutes. The resulting roadmap concentrated the nodes at regions where much detail was present in the environment while the open spaces were only sparsely covered. This resulted in a small roadmap.

While the RR technique outperforms the PRM and visibility PRM for these specific two- and -three dimensional problems, there are of course problems for which this does not hold. In particular, when the number of dimensions increases, the RR technique will be outperformed as the power of PRM techniques is that they are not sensitive to the dimensionality of the problem. Also, in many motion planning problems full coverage of the $\mathcal{C}$-space is not required as queries will be 'reasonable'. In such situations the preprocessing times for PRM and visibility PRM will be much lower. But the roadmaps they produce will still be large, leading to increased query times.



(a) Grid (b) Rotated grid



(c) Toy village

Fig. 6. The resulting roadmaps for the Grid, Rotated grid and Toy village environments

TABLE I

STATISTICS FOR THE FOUR ENVIRONMENTS

Environment	Grid resolution	Technique	Time (s)	$\|N\|$	$\|E\|$	$\|G\|$
Grid	$[80 \times 80]$	RR	0.33	13	12	88
		PRM	0.74	158	157	638
		vis-PRM	20.49	31	30	381
Rotated grid	$[200 \times 200]$	RR	6.78	249	248	388
		PRM	16.31	5889	5887	2328
		vis-PRM	>3600.00	n.a.	n.a.	n.a.
Manipulator	$[60 \times 60 \times 60]$	RR	10.81	313	306	380
		PRM	86.24	11095	11091	2306
		vis-PRM	>3600.00	n.a.	n.a.	n.a.
Toy village	$[180 \times 22 \times 160]$	RR	556.44	176	142	1268
		PRM	>3600.00	n.a.	n.a.	n.a.
		vis-PRM	>3600.00	n.a.	n.a.	n.a.

VI. CONCLUSION

We presented a new efficient algorithm that creates small roadmaps for two- and three-dimensional motion planning problems. The algorithm ensures that every valid query can be connected to the roadmap, as is required to solve the problem. If there exists a path in the (discretized) free space that connects the query, the algorithm ensures that a path can be found in the roadmap.

We compared our algorithm with PRM and visibility PRM. In our experiments, the Reachability Roadmap algorithm was faster than the other two techniques and produced considerably smaller roadmaps. While the PRM needed many nodes and connections until it solved the problem, the visibility PRM created small roadmaps but had great difficulties in getting the free space covered and connected. It was surprising to see how few nodes are actually required to build covering roadmaps.

We are currently investigating how to efficiently extend the technique to higher dimensions. A roadmap that is built for a lower dimensional subspace could be used to guide the motions for an object operating in a high-dimensional configuration space, see e.g. [22], [23]. We believe that this approach will enhance the quality of motion planners, in particular when query time is our major concern.

REFERENCES

[1] J.-C. Latombe, *Robot Motion Planning*. Kluwer, 1991.

[2] S. LaValle, *Planning Algorithms*. http://msl.cs.uiuc.edu/planning, 2005.

[3] T. Siméon, R. Chatila, and J.-P. Laumond, "Computer aided motion for logistics in nuclear plants," in *International symposium on artificial intelligence, robotics and human centered technology for nuclear applications*, 2002, pp. 46–53.

[4] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 995–1001.

[5] J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan, "A random sampling scheme for path planning," *International Journal of Robotics Research*, vol. 16, pp. 759–744, 1997.

[6] R. Geraerts and M. Overmars, "Sampling techniques for probabilistic roadmap planners," in *Conference on Intelligent Autonomous Systems*, 2004, pp. 600–609.

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

[8] D. Nieuwenhuisen, A. Kamphuis, M. Mooijekind, and M. Overmars, "Automatic construction of roadmaps for path planning in games," in *International Conference on Computer Games: Artificial Intelligence, Design and Education*, 2004, pp. 285–292.

[9] V. Boor, M. Overmars, and A. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *IEEE International Conference on Robotics and Automation*, 1999, pp. 1018–1023.

[10] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 4420–4426.

[11] J.-M. Lien, S. Thomas, and N. Amato, "A general framework for sampling on the medial axis of the free space," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 4439–4444.

[12] C. Nissoux, T. Siméon, and J.-P. Laumond, "Visibility based probabilistic roadmaps," in *IEEE International Conference on Intelligent Robots and Systems*, 1999, pp. 1316–1321.

[13] R. Geraerts and M. Overmars, "Reachability analysis of sampling based planners," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 406–412.

[14] J. O'Rourke, *Art Gallery Theorems and Algorithms*. New York: Oxford University Press, 1987.

[15] H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized Voronoi graph," *International Journal of Robotics Research*, vol. 19, no. 2, pp. 96–125, 2000.

[16] Y.-H. Lee, T.-W. Kao, and S.-S. Lee, "Optimal parallel algorithms for computing the chessboard distance transform and the medial axis transform on RAP," in *IEEE International Symposium on Parallel Architectures, Algorithms and Networks*, 1996, pp. 22–28.

[17] J. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[18] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press/ McGraw-Hill Book Company, 2001.

[19] F. Hwang, D. Richards, and P. Winter, *The Steiner Tree Problem*. North-Holland, 1992.

[20] G. van den Bergen, *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann, 2003.

[21] R. Geraerts, "http://www.cs.uu.nl/people/roland," 2005.

[22] J. van den Berg and M. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 453–460.

[23] M. Foskey, M. Garber, M. Lin, and D. Manocha, "A Voronoi-based hybrid motion planner," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 55–60.