

USING SPHERES TO IMPROVE
MOTION PLANNING ALGORITHMS

A Senior Honors Thesis

by

SHAWNA LYNN MILLER

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
in partial fulfillment of the requirements of the

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOWS

April 2001

Group: Computer Science

USING SPHERES TO IMPROVE
MOTION PLANNING ALGORITHMS

A Senior Honors Thesis

by

SHAWNA LYNN MILLER

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
in partial fulfillment for the designation of

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOW

Approved as to style and content by:

Nancy M. Amato
(Fellows Advisor)

Edward A. Funkhouser
(Executive Director)

April 2001

Group: Computer Science

ABSTRACT

Using Spheres to Improve
Motion Planning Algorithms. (April 2001)

Shawna Lynn Miller
Department of Computer Science
Texas A&M University

Fellows Advisor: Dr. Nancy M. Amato
Department of Computer Science

Motion planning algorithms aim to solve the problem of finding a collision-free path to move an object between start and goal positions in an environment. An environment consists of obstacles that the object must avoid. A useful abstraction is the object's configuration space (C-space). Each point in the C-space represents a unique position and orientation of the object in the environment. A valid point is a configuration where the object is not colliding with itself or an obstacle. The C-space consists of all possible positions and orientations, valid or not. Although the robot is simplified to a single point, the obstacles become very complicated in the C-space. It is too expensive to compute them.

Probabilistic roadmap methods are one type of method to solve motion planning problems. They use randomization to overcome this difficulty. The roadmap represents collision-free paths that the object can navigate. The basic probabilistic roadmap method generates nodes through uniform random sampling in the C-space. Variations of the probabilistic roadmap method use different methods to generate and connect roadmap nodes.

Probabilistic roadmaps work well in many applications, but in some situations they can be inefficient and unsuccessful. Our research provides an optimization of these algorithms using C-space spheres. Spheres, centered at each configuration in

the roadmap, define free areas of the C-space. The radius of a sphere is the node's C-space clearance.

Intersecting spheres identify good pairs of nodes for connection. Since an edge connecting the two nodes lies entirely inside both spheres, it is automatically known to be collision-free. C-space obstacles are not explicitly computed, so the C-space clearance must be approximated. Intersecting spheres no longer identify collision-free edges but now only likely edges. Verifying that the edge is valid can be postponed until query time. In practice, most edges identified by intersecting spheres are collision-free. This could dramatically reduce the running time of the algorithm.

These spheres also aid in expanding the roadmap. The roadmap is expanded by generating new nodes whose spheres intersect the spheres of existing nodes. This keeps the number of connected components small and gives good coverage of C-space.

ACKNOWLEDGMENTS

First of all, I would like to thank Dr. Nancy M. Amato, my advisor, for her encouragement. Thank you for showing me how exciting research can be, answering all my questions, and inspiring new ideas.

I would like to thank the members of the DSMFT (Dynamic Spatial Modelling for Tomorrow) group for their ideas, help, and insight in this work: Burchan Bayazit, Lucia K. Dale, Li Han, Jinsuck Kim, Sooyong Lee, Jyh-Ming Lien, Marco Morales, Ian Remmler, Wookho Son, Rick Stover, Sujay Sundaram, Guang Song, Daniel Vallejo, and Jennifer Walter.

I would also like to thank my family for their love and support in everything I do. Thank you for always encouraging me to do my best and loving me regardless of what that is.

Finally, I would like to thank Nathan Thomas, my fiancé. Thank you, Nathan, for encouraging me and supporting me, even when it means bringing me dinner when I'm up late working in the lab.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
ACKNOWLEDGEMENTS	v
TABLE OF CONTENTS	vi
LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTER	
I INTRODUCTION	1
II PREVIOUS WORK	2
A. Probabilistic Roadmap Method (<i>PRM</i>)	3
B. Obstacle-Based Probabilistic Roadmap Method (<i>OBPRM</i>)	4
C. Medial Axis Probabilistic Roadmap Method (<i>MAPRM</i>)	4
D. <i>Lazy PRM</i>	6
E. Customizable <i>PRM</i> (<i>C-PRM</i>)	7
III BASIC ALGORITHM	8
A. C-Space Spheres	8
B. Roadmap Expansion	10
IV EXPERIMENTAL RESULTS	13
A. Sandwich Environment	13
B. House Environment	14
C. City Environment	15
V ANALYSIS	17
A. Node Generation Methods	17
B. Pushing Methods	18
C. Approximate Edge Validation	18

CHAPTER	Page
	D. Spheres vs. Other <i>PRM</i> Algorithms 21
VI	CONCLUSIONS 22
	A. Summary of Results 22
	B. Future Work 22
	REFERENCES 24
	VITA 26

LIST OF TABLES

TABLE		Page
I	Sandwich Environment Results	14
II	House Environment Results	15
III	City Environment Results	16
IV	Comparison of Node Generation Methods	18
V	Finding Constrained Paths	19
VI	Algorithm Comparison on House Environment	21

LIST OF FIGURES

FIGURE		Page
1	<i>OBPRM</i>	4
2	<i>MAPRM</i>	5
3	Definition of C-space Sphere	9
4	Intersecting C-space Spheres	9
5	Approximate C-space Clearance	11
6	Roadmap Expansion	12
7	Sandwich Environment	13
8	House Environment	15
9	City Environment	16
10	Unconstrained Path	20
11	Constrained Path	20

CHAPTER I

INTRODUCTION

Automatic motion planning has been an expanding area of research over the past several years in computer science. In its simplest form, motion planning deals with finding a collision-free path to move an object between a start and a goal [1]. Although this is a very simple problem for humans, computers lack the intuition necessary to solve these problems easily [2]. Motion planning has many applications in robotics, computer animation, computer-aided design, virtual reality systems, and computational biology.

A complete motion planning algorithm exists that is guaranteed to find a collision-free path provided there is one, but it is computationally expensive and impractical for even the simplest applications [1]. Many other algorithms have been developed that are guaranteed to be fast but are not guaranteed to find a path. The goal of current research is to find an algorithm that provides a good balance between speed and completeness.

There are several different classes of motion planning algorithms. One class of algorithms in particular, the Probabilistic Roadmap Method and its variations, uses randomization to build a roadmap for the robot [3, 4, 5]. Once the roadmap is built, it can be used to find paths for multiple queries (pairs of start and goal configurations). Another advantage of these methods is that they are general. They can be applied to any type of robot and to any environment. These two characteristics make this class of algorithms very attractive.

The journal model is *IEEE Transactions on Automatic Control*.

CHAPTER II

PREVIOUS WORK

As the robot's degrees of freedom (dof) increase and the robot becomes more complex, so does the motion planning problem. A useful abstraction is the robot's configuration space (C-space) [6] whose dimension equals the robot's number of dof. A configuration consists of the robot's position and orientation in the environment and any other parameters required to completely describe the robot. For example, the configuration of a simple point robot in three-dimensional space has three dimensions: its position along the x-axis, y-axis, and z-axis. The configuration of a robot with volume has six dimensions: its position along the x-axis, y-axis, and z-axis and its rotation about each axis. For an articulated robot the dimension increases further. For example, a robot with 4 links has a 9-dimensional configuration: the position along the x-axis, y-axis, and z-axis, the rotation about each axis for the base link, and three joint angles for the remaining three links. Clearly, as the robot becomes more complex, it requires more parameters.

A configuration is considered collision-free or valid if the robot is not in-collision with itself or any other obstacle in the environment. Otherwise, the configuration is considered invalid. The C-space consists of all possible configurations, valid or not.

C-space abstracts the robot down to a single point but transforms the obstacles into very complicated objects in C-space. The C-obstacles are so complex that it would be computationally expensive and impractical to compute them. This, coupled with the high-dimensionality of C-space, makes a complete solution to the motion planning problem slow and impractical. Many solutions use randomization to overcome this difficulty.

A. Probabilistic Roadmap Method (*PRM*)

Some of the most promising algorithms use random sampling to generate a roadmap in C-space. Probabilistic Roadmap Methods (*PRMs*) were developed independently by Kavraki and Latombe at Stanford University and Overmars and Svestka at Utrecht University in the Netherlands [3, 4, 5]. The roadmap is a graph in the free regions of the C-space. Each node in the graph corresponds to a valid configuration of the robot in the workspace. Each edge connecting two nodes represents a collision-free path between the two nodes.

Once a roadmap is created, the motion planning problem decomposes into a simpler one of connecting the start and the goal configurations to the roadmap and searching the roadmap for a path between them. As long as they are both connected by edges to the same connected component of the roadmap, a path can easily be found. If they are not, the algorithm fails to find a path.

PRMs are built in two phases: node generation and connection. The basic *PRM* algorithm generates nodes uniformly at random in C-space. Edges are connected by checking the node's k closest neighbors. If there exists a valid path, the edge is added to the roadmap.

The basic *PRM* algorithm provides a uniform distribution of nodes quickly, but it struggles to find nodes in narrow passages. Because nodes are randomly generated, the probability that *PRM* will generate nodes in narrow passages is small. *PRMs* are good at finding paths very quickly, but tend to fail when the path must pass through a narrow passage.

B. Obstacle-Based Probabilistic Roadmap Method (*OBPRM*)

In order to overcome this weakness of *PRMs*, Obstacle-Based Probabilistic Roadmap Method (*OBPRM*), developed by Amato and others at Texas A&M University [7, 8], generates nodes around the surface of obstacles. The resulting roadmap is densest where narrow corridors typically lie, see Figure 1. The key to the success of this method is a good distribution of nodes around each obstacle in C-space. Since the C-obstacles are never explicitly computed, this is impossible to guarantee and difficult to achieve. Despite this shortcoming, *OBPRM* performs surprisingly well and can solve many difficult problems with narrow passages.

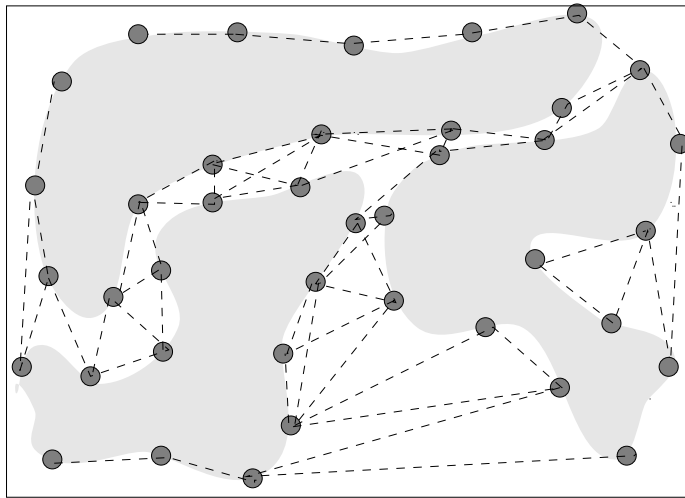


Fig. 1. *OBPRM* generates nodes near the C-obstacle surfaces to obtain nodes in narrow passages.

C. Medial Axis Probabilistic Roadmap Method (*MAPRM*)

Another variation proposed by Wilmarth, Amato and Stiller at Texas A&M University also biases node generation. Medial Axis Probabilistic Roadmap Method (*MAPRM*) [9, 10] randomly generates configurations and pushes them onto the me-

dial axis of the free C-space, see Figure 2. A configuration has maximum clearance when the distances to the closest 2 (or more) C-obstacles are equal. The medial axis is the set of all such configurations. By pushing them onto the medial axis, roadmap nodes lie along the center of passages. The key strength of this method is that it uses every configuration generated, free and in-collision. Free nodes are pushed directly to the medial axis. In-collision nodes are first pushed out of the C-obstacle towards the nearest free space boundary and then pushed to the medial axis. *MAPRM* generates nodes in narrow passages with a higher probability than random sampling. The weakness of this method is that it does not guarantee the connectivity of the roadmap. Like the basic *PRM* it attempts to connect each node with nearby ones.

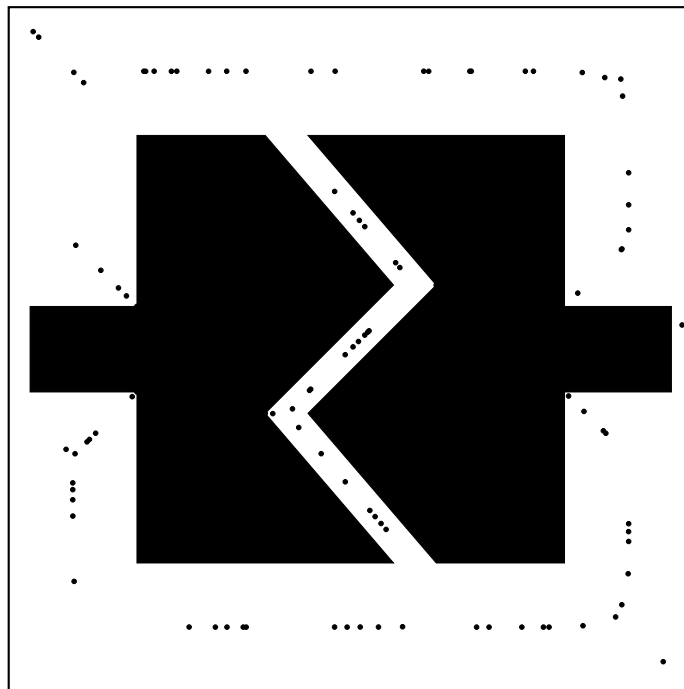


Fig. 2. *MAPRM* generates nodes on the medial axis of the free space to bias the sampling towards narrow passages.

D. *Lazy PRM*

Most of the time spent in building a roadmap is consumed in the connection phase. In fact, the connection phase can take up to 98% of the total running time [7, 3]. The motivation of *Lazy PRM* [11] is to reduce the time spent in the connection phase and thereby increase the efficiency of the algorithm. Like *PRM* nodes are generated uniformly at random. *Lazy PRM* then assumes that all nodes and all possible edges are valid and adds them to the roadmap. Since the nodes and edges are not checked for collision, the running time dramatically drops. In the query phase, the start and goal configurations are connected to the roadmap, and a shortest path is extracted. Each node and edge in the path is then checked for collision. If a node or edge is found invalid (or in-collision), it is thrown away and a new shortest path is extracted. This process repeats until either a collision-free path is found, or the start and goal configurations are no longer connected through the roadmap.

A similar approach, *Fuzzy PRM*, was independently proposed by Nielsen and Kavraki [12]. It only adds valid nodes to the roadmap and adds all possible edges. Instead of initially assuming all edges are collision-free, as *Lazy PRM*, *Fuzzy PRM* assigns each edge a weight. This weight corresponds to the probability that the edge is collision-free. The probability is based on the distance between the two configurations. During the query phase, the path with the highest probability (instead of the shortest path) is extracted. Each edge along that path is placed in a priority queue. The edge with the lowest probability is checked at a slightly higher resolution. If the edge is found to be in-collision, it is thrown away and a new path is extracted. If the edge is not found to be in-collision, it is inserted back into the priority queue with a new (higher) probability. This process repeats until a valid path is found, or the start and goal are no longer connected through the roadmap.

E. Customizable *PRM* (*C-PRM*)

Customizable *PRM* (*C-PRM*) builds a coarse roadmap by only partially checking the edges [13]. Like *Lazy PRM* and *Fuzzy PRM*, it postpones complete edge validation until the query phase. The advantage of our method over *Lazy PRM* and *Fuzzy PRM* is that the roadmap is more representative of the true roadmap, since it partially validates edges during the construction phase. This even holds true for coarse validations like simply checking the midpoint.

The main contribution of our method is the roadmap can be customized for multiple requirements/constraints. These constraints include maintaining a specified clearance, minimizing the rotation about a certain axis, or restricting the number of sharp turns. Other constraints specific to the application can be added to the basic framework easily. Our approach produces efficient results in a wide range of problems. Because the path can be restricted to meet certain constraints, *C-PRM* makes *PRM* a more feasible solution in practical applications.

CHAPTER III

BASIC ALGORITHM

The basic *PRM* has difficulty finding a solution when the path must pass through a narrow passage. Variations like *OBPRM* and *MAPRM* find more nodes in narrow passages, but the roadmap may contain many connected components. Our algorithm attempts to use the strength of *OBPRM* and *MAPRM* to find key nodes while maintaining the connectivity of the roadmap. The algorithm generates several seed nodes, either by *OBPRM* or *MAPRM*, and “grows” the nodes into a roadmap. As the roadmap grows, the validity of the edges is approximated by C-space spheres. This algorithm applies the *C-PRM* philosophy by postponing complete edge validation until the query phase. The user may also impose certain requirements/constraints on the solution path as implemented by *C-PRM*.

A. C-Space Spheres

Each configuration has a clearance in the C-space, the distance to the closest C-obstacle. This clearance, c , defines a sphere, with radius c , centered at the given configuration, see Figure 3. Because the radius is the minimum distance to any C-obstacle, everything inside the sphere is automatically known to be collision-free. The sphere can be thought of as a bubble of free configurations in the C-space. When two spheres intersect, a straight line connecting their centers is entirely contained inside the union of both spheres. Thus, C-space spheres provide for automatic node connection: If the spheres intersect, a straight-line edge between the two centers is known to be collision-free and can be added to the roadmap without checking each point along the edge for collision, see Figure 4.

Because obstacles in the C-space are complicated and expensive to compute, a

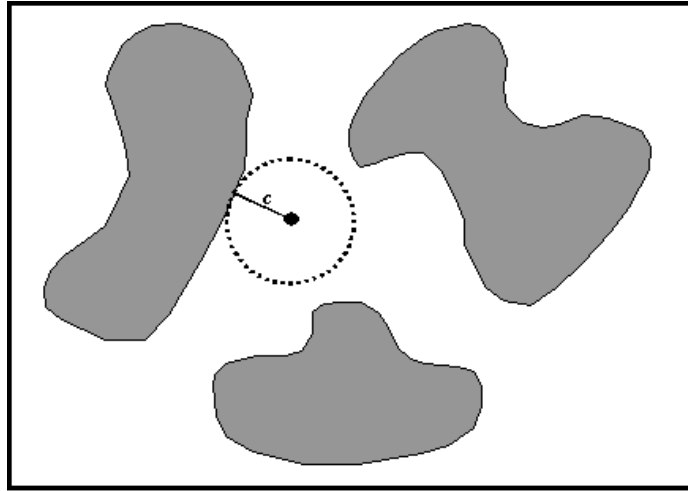


Fig. 3. A C-space sphere is centered at the node with radius c , where c is the node's C-space clearance.

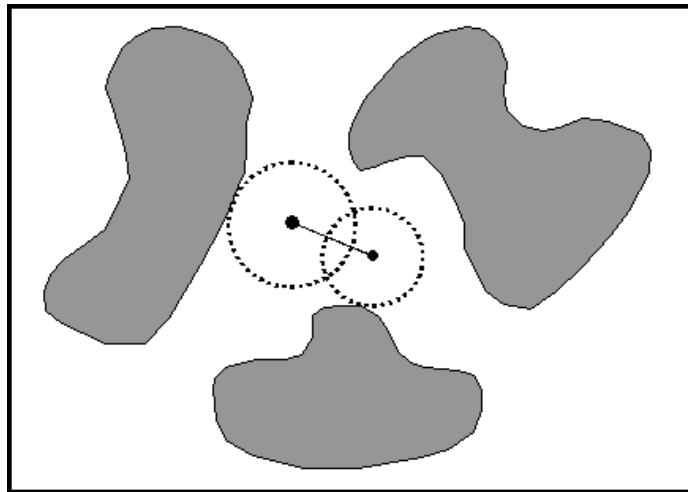


Fig. 4. When two C-space spheres intersect, a straight line edge between them is automatically known to be collision-free.

configuration’s exact clearance cannot be easily determined. Although calculating the actual clearance is time consuming and impractical, approximating the clearance is not. This approximate clearance can still be used to define C-space spheres. Instead of intersecting spheres indentifying automatic connections, they identify “good” candidates for connection. This way, only those pairs of configurations that have a good probability of being connected are attempted. This reduces the number of calls to the collision detection package by weeding out unlikely pairs. This is similar in approach to *Lazy PRM* and *Fuzzy PRM*, except approximating edge validity is different.¹

To approximate the actual C-space clearance, we compute the C-space clearance in k random directions, see Figure 5. The smallest clearance is then assumed to be the actual C-space clearance. Although the clearance is only an approximation, experiments show that it is a good one. In particular, we have found accurate approximations using as few as 3 or 5 directions.

B. Roadmap Expansion

One concern of most *PRMs* is connectivity. The roadmap must represent the connectivity of the C-space for the algorithm to perform well. To maintain connectivity in the roadmap, the existing roadmap is expanded or grown out into the C-space. In order to keep the roadmap from becoming too dense, only new nodes, or nodes that have not been expanded yet, are expanded.

¹*Lazy PRM* does not approximate edge validity and initially assumes all edges to be collision-free. *Fuzzy PRM* approximates edge validity through probabilities.

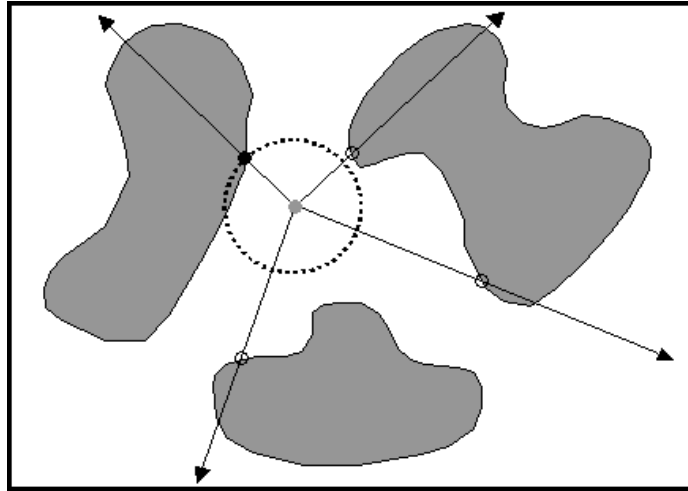


Fig. 5. To approximate the C-space clearance, several rays are chosen at random. Then, the clearance along each ray is computed. The smallest clearance is taken to be the approximate C-space clearance of the node.

To expand a node, first a direction is selected. This may be done in several different ways:

- Select a direction at random.
- Select a direction away from the nearest C-obstacle.
- Compute the average direction of the k closest neighbors and select a direction opposite to it.

Then a new node is placed on the surface of the C-space sphere in the selected direction. This new node is pushed outwards in one of two ways:

- The simplest method pushes the new node outwards along the generated direction until just before the spheres are no longer intersecting.
- The second method pushes the node to the medial axis.

Thus, the new nodes have maximum clearance and are more likely to intersect other spheres.

Pushing the new nodes outward allows for maximum coverage of the C-space while maintaining connectivity. For each node, several new nodes may be pushed outward. This process is repeated several times until the roadmap adequately fills the free C-space, see Figure 6.

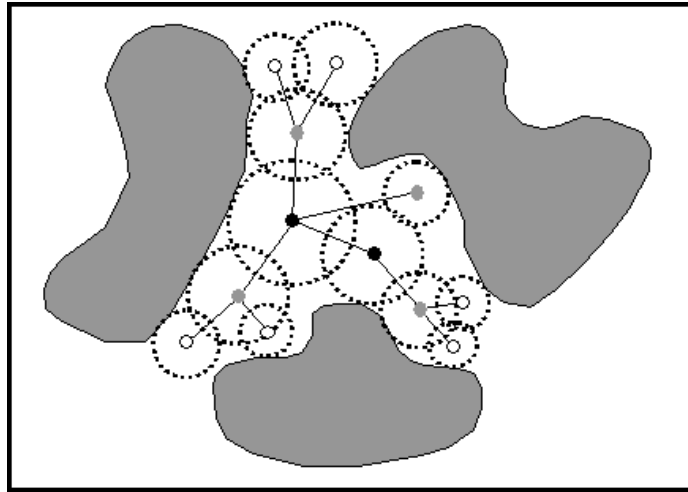


Fig. 6. The seed nodes are shown in black. After the first level of expansion, the gray nodes are added to the roadmap. After the second level of expansion, the white nodes are added to the roadmap. As the roadmap expands, it fills the free regions of the C-space.

CHAPTER IV

EXPERIMENTAL RESULTS

We will study several representative environments. We will then use the results from these environments to show proof of concept.

A. Sandwich Environment

The first environment consists of two parallel plates, and the robot is a small cube (Figure 7). The goal is to move the robot out from inbetween the two plates. This is a simple environment because it is not cluttered and the robot is a simple rigid body.

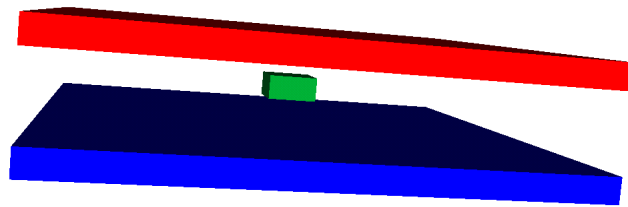


Fig. 7.

We ran our algorithm starting with 20 seed nodes. The roadmap went through 3 levels of expansion, and during each level we expanded each new node twice. In all our experiments, we chose to expand the node in the opposite direction of the nearest C-obstacle. Preliminary testing showed that a purely random direction was ineffective. We have not yet implemented picking a direction away from the k closest neighbors.

We expanded the roadmap with two pushing methods: straight-line (Line) and medial axis (MA). We also used two different values for n , the number of directions to check when approximating the C-space clearance, namely 3 and 7. In previous testing, we found that $n < 3$ produced inaccurate approximations and $n > 10$ was inefficient. Finally, we implemented the algorithm in two different ways:

- Validate all edges before adding them to the roadmap (like most *PRM* methods)
- Postpone complete edge validation until the query phase (like *Lazy PRM*, *Fuzzy PRM*, and *C-PRM*).

The results are given in Table I.

Table I. Sandwich Environment Results

Seed Nodes	Pushing Method	n	Check Edges	Nodes	Edges	CC's	Running Time(sec)	Solved Query
20	MA	3	Y	110	120	2	2.88	Y
20	MA	3	N	110	120	2	2.97	Y
20	MA	7	Y	156	174	4	9.15	Y
20	MA	7	N	156	175	3	8.88	Y
20	Line	3	Y	131	139	4	2.27	Y
20	Line	3	N	131	141	2	2.17	Y
20	Line	7	Y	210	230	2	6.63	Y
20	Line	7	N	210	230	2	6.43	Y

B. House Environment

This environment consists of a house and a table (Figure 8). The goal is to move the table from outside the house to inside one of the rooms. This environment is still uncluttered, but now the robot (table) must fit through a smaller passage.

Testing was done similarly to Section A above. This time we started the roadmap with 50 seed nodes instead of 20. The results are given below in Table II.

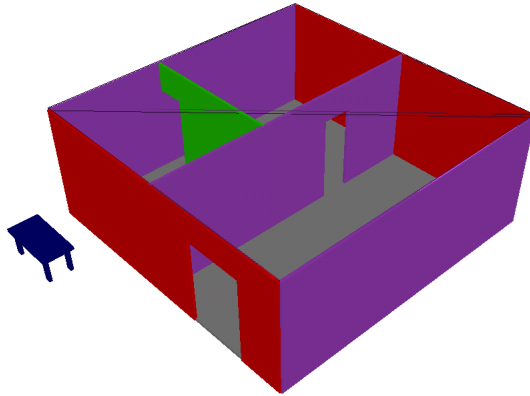


Fig. 8.

Table II. House Environment Results

Seed Nodes	Pushing Method	n	Check Edges	Nodes	Edges	CC's	Running Time(sec)	Solved Query
50	MA	3	Y	522	693	3	33.81	Y
50	MA	3	N	522	701	2	32.91	N
50	MA	7	Y	640	739	5	91.71	N
50	MA	7	N	640	739	5	87.64	N
50	Line	3	Y	529	692	11	22.09	Y
50	Line	3	N	333	707	6	21.52	N
50	Line	7	Y	616	714	6	50.09	N
50	Line	7	N	616	717	3	49.91	N

C. City Environment

The city environment¹ consists of several buildings, and the robot is a helicopter (Figure 9). This environment is more cluttered than the previous two, and the robot is slightly more complicated. These two factors led to the longer running times. The results are given in Table III.

¹This environment was provided by LAAS-CNRS in Toulouse, France.

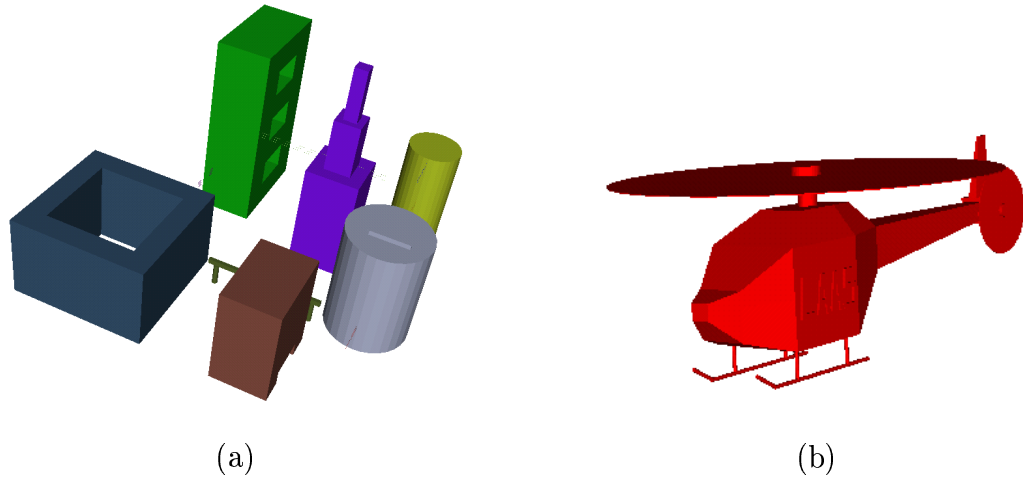


Fig. 9.

Table III. City Environment Results

Seed Nodes	Pushing Method	n	Check Edges	Nodes	Edges	CC's	Running Time(sec)	Solved Query
20	MA	3	Y	222	265	2	84.75	Y
20	MA	3	N	222	270	1	82.84	Y
20	MA	7	Y	224	236	1	218.55	Y
20	MA	7	N	224	236	1	212.67	Y
20	Line	3	Y	242	280	7	91.28	Y
20	Line	3	N	242	290	1	91.41	Y
20	Line	7	Y	247	257	3	175.51	Y
20	Line	7	N	247	258	2	178.41	N

CHAPTER V

ANALYSIS

In this section we analyze and interpret the experimental results described in Chapter IV and some additional experiments described here to study particular issues.

A. Node Generation Methods

In our algorithm, the seed nodes can be generated by many different methods. The node generation method greatly affects the quality of the resulting roadmap. Consider the house environment. As before, we want to move the table from outside the house to inside one of the rooms. We ran the algorithm several different times. Each time, we began the roadmap with 50 seed nodes generated by *PRM*, *OBPRM*, or *MAPRM*. Table IV clearly shows that roadmap quality heavily depends upon the quality of the seed nodes.

Given the same initial parameters (i.e., same number of seed nodes, pushing method) *PRM* and *MAPRM* were able to solve the query, but *OBPRM* was not. Since *OBPRM* generates nodes close to the surfaces of C-obstacles, the corresponding spheres will be very small. The roadmap was not able to expand out into the C-space very far. Since it poorly represented the connectivity of the C-space, it was unable to solve the query.

Although both *PRM* and *MAPRM* were successful, the quality of the roadmap built by *MAPRM* seed nodes is better. First, *MAPRM* found more edges than *PRM*. It also did this in a shorter amount of time. Since the seed nodes lie on the medial axis, their clearances (and spheres) are larger. This allows *MAPRM* to expand farther into the C-space faster than *PRM*. Clearly, *MAPRM* is the best choice for most situations.

Table IV. Comparison of Node Generation Methods

Generation Method	Pushing Method	Nodes	Edges	CC's	Running Time(sec)	Solved Query
<i>PRM</i>	MA	505	610	4	34.4	Y
<i>PRM</i>	Line	513	615	7	24.91	Y
<i>OBPRM</i>	MA	502	526	9	28.2	N
<i>OBPRM</i>	Line	539	545	27	21.02	N
<i>MAPRM</i>	MA	522	693	3	33.81	Y
<i>MAPRM</i>	Line	529	693	11	22.09	Y

B. Pushing Methods

Different pushing methods also effect the roadmap. In all our experiments, we used two pushing methods: straight-line (Line) and medial axis (MA). Straight-line is a more naive approach; it is quick to compute but not very intelligent. Medial axis, on the other hand, is more intelligent, but slower to compute. The effects of these methods can be seen in Tables I, II, and III given previously in Chapter IV.

We would expect the MA push to be better since the nodes will lie on the medial axis. However, the new spheres may not connect to the seed sphere any longer. Our results are inconclusive at this point but we believe the MA push should perform better if used properly. We will investigate this in future work.

C. Approximate Edge Validation

Postponing complete edge validation until the query phase tends to increase the number of edges, decrease the number of connected components, and reduce the time to build a roadmap (see Chapter IV, Tables I, II, and III). The speed-up between the two techniques was not as much as initially expected. Further investigation showed that the bottleneck of the algorithm is in computing the approximate C-

space clearance. This is computed many times when building the roadmap (more than any other function), therefore it dominates the running time. By optimizing this function, the running time for the algorithm would dramatically decrease. This will be a focus of future research.

We also applied the *C-PRM* approach to the city environment (Section C). We constrained the helicopter’s rotation about the x and y axes to produce a more realistic path. We allowed a maximum tilt of 90° from horizontal. We applied this approach to several different *PRM* algorithms: *PRM*, *MAPRM*, and our algorithm. *OBPRM* could not build a successful roadmap. The results are shown in Table V.

Table V. Finding Constrained Paths

Algorithm	Nodes	Edges	CC’s	Running Time(sec)	Solved Query
Spheres	1567	3924	2	887.92	Y
<i>PRM</i>	5000	4954	46	3129.77	N
<i>MAPRM</i>	4879	4835	44	3746.61	N

Our algorithm was the only one that could solve the query. *PRM* and *MAPRM* were first run with 5000 nodes. We doubled the number of nodes and allowed them to run overnight. Both algorithms could not finish building a roadmap in this amount of time. Given enough time, they could finish building the roadmap and possibly solve the query, but these results would clearly be inefficient and impractical. Our algorithm was the only one that could find a successful path in a reasonable amount of time. Also note, that the same roadmap that our algorithm generated was able to solve multiple queries in a constrained situation.

Figures 10 and 11 show snapshots from the paths found by our algorithm. The unconstrained path (Figure 10) required the helicopter to fly upside-down, very unrealistic. The constrained path (Figure 11) produced much more realistic results.

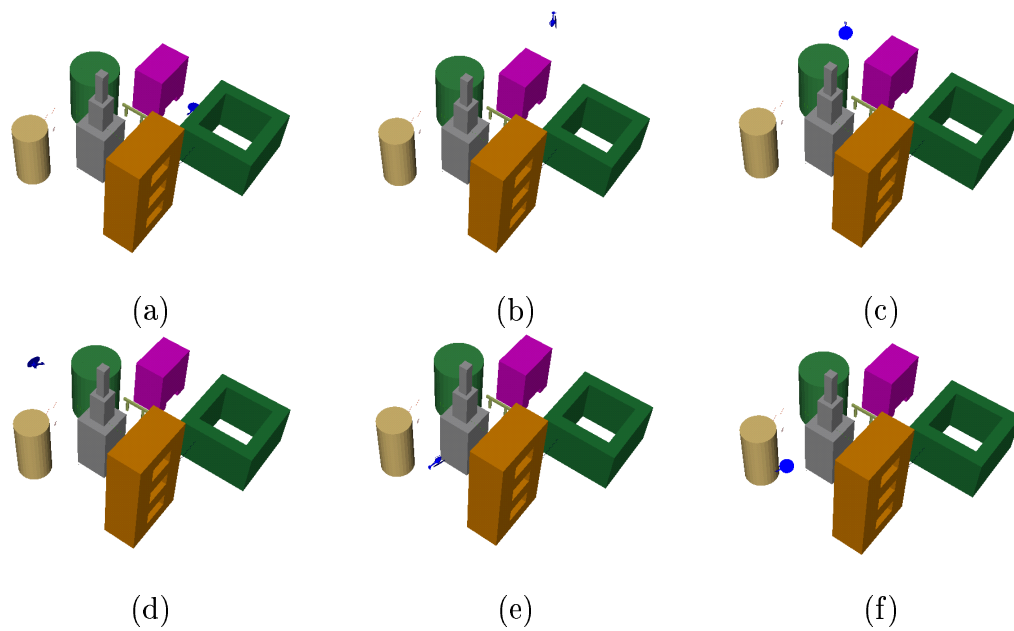


Fig. 10. Snapshots of the path found for the helicopter flying in the city environment when its orientation was not restricted. In (b) and (d), the helicopter's tilt is greater than 90° .

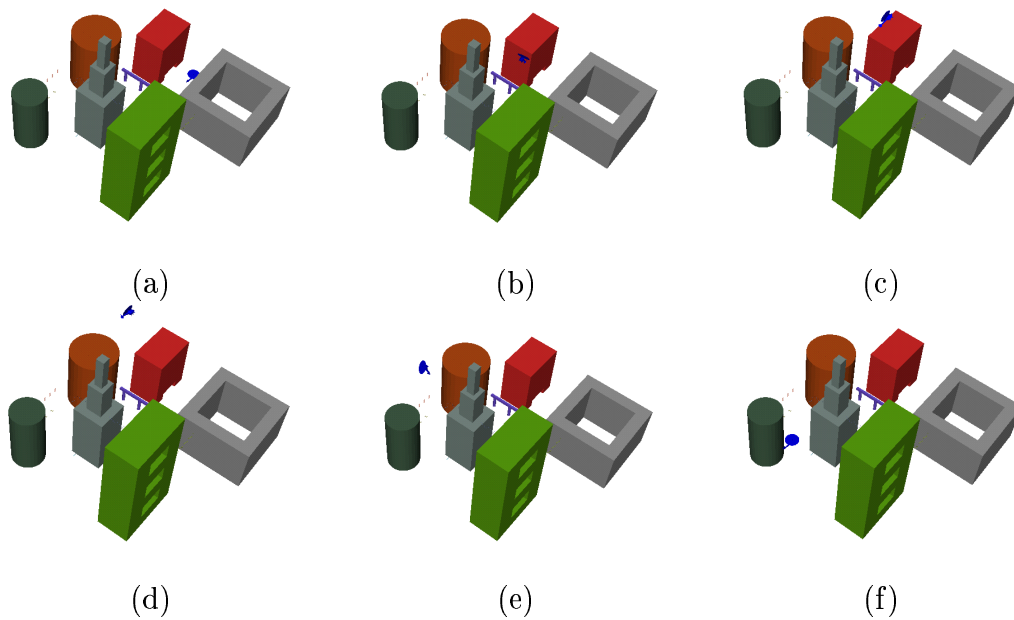


Fig. 11. Snapshots of the path found for the helicopter flying in the city environment when its orientation was restricted. In all cases, the helicopter's tilt is less than 90° .

D. Spheres vs. Other *PRM* Algorithms

Again, we used the house environment to compare our algorithm with other *PRMs*. For each algorithm, we built the smallest roadmap possible that would still solve the query¹. The results are given in Table VI.

Table VI. Algorithm Comparison on House Environment

Method	Nodes	Edges	CC's	Running Time(sec)
PRM	700	696	4	17.32
OBPRM	522	508	14	11.4
MAPRM	20	18	2	0.81
Spheres	28	26	2	2.21

PRM required the largest roadmap at 700 nodes. This is because random sampling is not biased towards narrow passages like the other algorithms. *OBPRM* required a smaller roadmap, 522 nodes, but threw away 978 nodes to generate them. Our algorithm was able to solve the query using 28 nodes. *MAPRM* required only 28 nodes to solve the query. It also built the roadmap in the shortest amount of time.

¹The query is the same as before: move the table from outside the house to inside one of the rooms.

CHAPTER VI

CONCLUSIONS

We designed and implemented a new method for building probabilistic roadmaps. It is based on spheres approximating the clearance in C-space. It uses C-space spheres to approximate edge validation and expand the roadmap.

Intersecting spheres identify “good” candidates for edge connection, and they are added to the roadmap as approximate edges. Then new spheres are generated near current spheres and pushed outward. Thus, the roadmap expands out into C-space while maintaining connectivity.

A. Summary of Results

Our solution is general and can be applied to many situations. The method itself is also general. Seed nodes are created by many different node generation techniques (*PRM*, *OBPRM*, *MAPRM*, etc.). As new generation techniques are developed they can be easily added to our basic framework. Different pushing methods (like straight line and medial axis) can also be easily added to improve roadmap quality.

Experimental results show our algorithm’s potential. There are some cases, specifically finding constrained paths, where our algorithm outperforms existing algorithms. Our algorithm was able to find solution paths when other algorithms failed to do so in a reasonable amount of time.

B. Future Work

One weakness of our algorithm is speed. In order to reduce the running times, we would like to further optimize the approximate C-space clearance computation. Since

this computation dominates the running time, speeding it up would greatly improve the algorithm's total speed.

As mentioned in Chapter III, Section B, only two expansion directions are currently implemented: selecting a random direction and selecting a direction opposite of the nearest C-obstacle. We would like to implement the third option, selecting a direction away from the nearest neighbors. This would keep the roadmap from becoming dense and encourage the roadmap to expand outward into unexplored regions of the C-space.

REFERENCES

- [1] J. C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.
- [2] J. Reif, “Complexity of the piano mover’s problem and generalizations,” in *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, 1979, pp. 421–427.
- [3] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high dimensional configuration spaces,” *IEEE Trans. Robot. Autom.*, vol. 12, pp. 566–580, 1996.
- [4] M. Overmars and P. Svestka, “A probabilistic learning approach to motion planning,” in *Proc. Workshop on Algorithmic Foundations of Robotics*, 1994, pp. 19–37.
- [5] L. Kavraki and J. C. Latombe, “Randomized preprocessing of configuration space for fast path planning,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1994, pp. 2138–2145.
- [6] T. Lozano-Pérez, “Spatial planning: A configuration space approach,” *IEEE Trans. Comput.*, vol. C-32, pp. 108–120, 1983.
- [7] N. M. Amato and Y. Wu, “A randomized roadmap method for path and manipulation planning,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1996, pp. 113–120.
- [8] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo, “OBPRM: An obstacle-based PRM for 3D workspaces,” in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998, pp. 155–168.

- [9] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, “MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999, pp. 1024–1031.
- [10] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, “Motion planning for a rigid body using random networks on the medial axis of the free space,” in *Proc. ACM Symp. on Computational Geometry (SoCG)*, 1999, pp. 173–180.
- [11] R. Bohlin and L. E. Kavraki, “Path planning using lazy prm,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 521–528.
- [12] C. L. Nielsen and L. E. Kavraki, “A two level fuzzy prm for manipulation planning,” Tech. Rep. TR2000-365, Computer Science, Rice University, Houston, TX, 2000.
- [13] G. Song, S.L. Miller, and N. M. Amato, “Customizing prm roadmaps at query time,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2001, To appear.

VITA

Shawna Lynn Miller began her undergraduate studies in the Fall of 1998. She is pursuing a Computer Engineering (Computer Science Track) Degree from the Department of Computer Science at Texas A&M University.

She works as an undergraduate researcher with the Dynamic Spatial Modelling for Tomorrow (DSMFT) group. DSMFT is a robotics group that specializes in motion planning algorithms, specifically *PRMs* and their variations.

She is interested in all aspects of robotics, from motion planning, to artificial intelligence systems, to the mechanics behind robot design. She is also interested in computer graphics and animation.

She plans to pursue a Masters of Computer Science after graduating from Texas A&M University in December, 2001.

Permanent address:

11462 Stoney Falls Drive
Houston, TX 77095

The typist for this thesis was Shawna Lynn Miller.