Randomized Motion Planning for Car-like Robots with C-PRM*

Guang Song Nancy M. Amato
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112
{gsong,amato}@cs.tamu.edu

Abstract

In this work, we propose a new approach for motion planning for nonholonomic car-like robots which is based on a Customizable PRM *(*C-PRM*). A major advantage of our approach is that it enables the* same *roadmap to be efficiently utilized for car-like robots with different turning radii, which need not be known before query time. Our* C-PRM*-based approach first builds a so-called control roadmap which does not incorporate any nonholonomic constraints. The control roadmap is used to efficiently generate 'good' configurations of the car, e.g., aligned with the roadway. The control roadmap is also used to guide the roadmap connection. The paths encoded in the roadmap consist of straight-line segments and arcs, where transitions between the two require full stops of the car. The control roadmap assists in the optimization and smoothing of these paths using cubic B-splines. Results with a simple car-like robot are very promising.*

1 Introduction

Automatic motion planning deals with finding a feasible sequence of motions to take some movable object (the 'robot') from a given initial configuration to some specified goal configuration. While complete motion planning algorithms do exist, they are rarely used in practice since they are computationally infeasible in all but the simplest cases [14, 19]. For this reason, attention has focused on probabilistic methods, which sacrifice completeness in favor of computational feasibility and applicability.

In particular, we note a class of algorithms known as *probabilistic roadmap (*PRM*)* methods. The idea behind PRMs is to create a graph of randomly generated collision-free configurations which are connected by a simple and fast local planning method. Actual global planning (queries) is carried out on this graph. The initial PRMs were very successful in solving difficult problems in high-dimensional configuration spaces (C-space) that had previously resisted efficient solution [12], and moreover, were simple and easy to implement, requiring only collision detection as a primitive operation. These successes led to the application of PRM motion planning techniques to a number of challenging problems arising in a variety of fields including robotics (e.g., closed-chain systems [10, 16]), CAD (e.g., assembly [25], maintainability [4, 8], deformable objects [2, 11]), and even computational Biology and Chemistry (e.g., ligand docking [3, 5, 20], protein folding [3, 22, 23]). Indeed, it can be argued that the PRM framework was instrumental in this broadening of the range of applicability of motion planning to include these kinds of problems.

Customizable PRM (C-PRM). In this work, we apply the PRM methodology to planning trajectories for car-like robots. Our approach is based on a PRM variant, called C-PRM (or Customizable PRM [21]), recently developed in our group. C-PRM differs from traditional PRMs in two ways. First, in the roadmap construction stage, we build coarse roadmaps by performing only approximate validation of roadmap nodes and edges. For example, we may approximately check the validity of an edge by only checking a small fraction of its intermediate configurations. Second, in the query stage, the roadmap is validated and refined only in the necessary regions, and moreover, is *customized* in accordance with any specified query preferences.

The advantages of C-PRM over traditional PRMs, which perform complete roadmap validation during preprocessing, are efficiency, flexibility, and adaptability for different robots. A C-PRM roadmap can be naturally pruned to satisfy different query requirements, and is iteratively refined in the meaningful areas as queries proceed. For car-like robots, this means that the roadmap we construct can be easily adapted and used for the planning of robots with different turning radii, which was not possible with other methods.

Motion planning for car-like robots. Car-like robots typically have three degrees of freedom, i.e., two positional dofs and one orientational dof, i.e. (x, y, θ), which are related through an intrinsic nonholonomic constraint:

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \tag{1}$$

whose intuitive meaning is that car-like robots can only move forward or backward. In addition, they generally also have a lower-bound on their turning radius. Therefore, a car-like robot cannot follow all collision free paths in a two-

*This research supported in part by NSF CAREER Award CCR-9624315, NSF Grants IIS-9619850, ACI-9872126, EIA-9975018, EIA-9805823, and EIA-9810937, and by the Texas Higher Education Coordinating Board under grant ARP-036327-017.

dimensional environment. The path has to curve 'slowly' enough for the car to follow.

Much research has been done on motion planning for nonholonomic car-like robots (see [9] for a review). While much of this work has used randomized methods, most of it has utilized potential field methods (e.g. [6, 13]) as opposed to the increasingly popular PRM technique. A notable exception is the work of Švestka and Overmars on car-like and tractor-trailer robots using their *PPP* (Probabilistic Path Planning) algorithm, see [9] and references therein. Their method first generates configurations in C-space ($R^2 \times [0, 2\pi)$, and then connects them using an RTR path local planning method. An RTR path is defined as the concatenation of a rotational path, a translational path, and another rotational path [26]. An alternative is to use a local planner that constructs the shortest path connecting the two configurations as was done in [18, 24]. Another randomized strategy that has been used for non-holonomic planning is the RRT (Rapid-Exploring Random Tree) approach [15]. Here, the the nonholonomic constraints (e.g., the turning radius) are used to construct a tree of feasible paths emanating from the starting configuration.

However, in all these previous methods, the roadmap is built specifically for robots with a particular, pre-defined turning radius. Thus, if different robots (with different turning radii) are to operate in the same environment, they would all need different roadmaps.

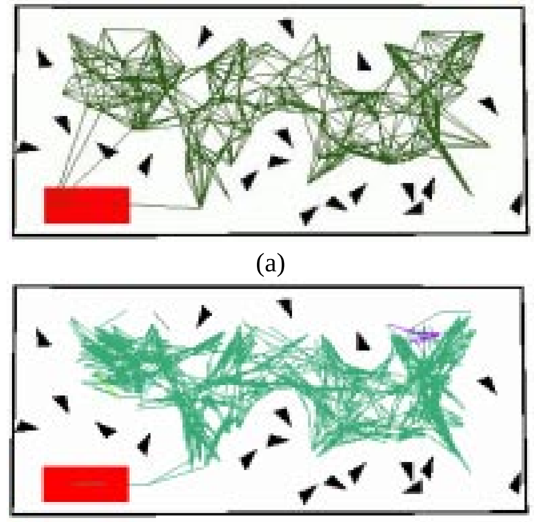
2 A C-PRM for Car-Like Robots

We propose a novel method for motion planning for nonholonomic car-like robots. A major advantage of our C-PRM-based approach is that it enables the *same* roadmap to be efficiently utilized for car-like robots with different turning radii. In addition, the computation costs (including roadmap construction and querying), even for a single car-like robot, are generally significantly less than with other techniques, such as the PPP [9] or RRT [15] methods.

However, this work is not just simply another application of C-PRM. In particular, the nonholonomic constraints, and our goal of supporting different turning radii with the same approximate roadmap, means that extra care must be taken when constructing the roadmap. For example, sampling strategies must be devised that are sensitive to the car's orientation, and connection (local planning) must be performed without knowledge of the car's turning radius (since it will not be known until query time). Thus, the roadmap is constructed hierarchically.

- First, we construct a *control roadmap* for quickly estimating the connectivity of the free space. Here, no restriction is placed on the car's orientation or turning radius and edges in the control roadmap are validated very coarsely. (See Fig. 1(a).)

- Next, an *approximate roadmap* is constructed from the control roadmap. Nodes correspond to the *midpoints* of the edges in the control roadmap, and they are now oriented along the direction of the edge (aligned with the "roadway"). Roadmap nodes are connected if they correspond to adjacent edges in the control roadmap. At this point, only a very coarse bound is placed on the turning radius – the turning radius of the actual robot will be enforced at the query stage. (See Figure 1(b).)



(a)

(b)

Figure 1: The (a) control and (b) approximate roadmaps.

As in [21], we believe the main strength of the roadmap is that it can provide an estimation of the free space and its connectivity. Even very coarse roadmaps, such as our control roadmap, can accomplish this task swiftly. They can then be refined as necessary using more detailed validations – but only in the regions determined promising by the initial coarse processing, which can save a significant amount of processing time. A similar philosophy is proposed in the Lazy PRM [7] and Fuzzy PRM [17] methods, which advocate an even coarser validation than C-PRM – either no validation at all [7] (i.e., accepting all roadmap nodes and edges in the construction phase) or validating nodes but not edges [17]. Our experience is that some, even very limited, validation of both nodes and edges provides fairly good estimates of the free space to guide path selection, which in turn can significantly reduce query costs.

In the following, we describe the details of the roadmap construction and the adaptable query process for enforcing variable turning radii. We model the car-like robot as a polygon moving in a two-dimensional environment, R^2, and its C-space is represented by $R^2 \times [0, 2\pi)$.

2.1 Control Roadmap

The *control roadmap* (Cntl-Rdmp) is constructed similarly to most PRM roadmaps, but using the C-PRM philosophy of postponing detailed validation. In particular, we first generate a sample of collision free nodes in the environment. As orientation information is not necessarily needed at this stage, for collision detection purposes we can use a disk of diameter d as our robot, where d is less than the dimensions of the car-like robot. These points are called *control points (CPs)*. Alternatively, a robot more similarly dimensioned to the car can be used. We use this second option since our 'cars' were very simple objects.

We then try to connect each CP to its k closest neighbors. An edge is added to the control roadmap if the midpoint of that edge is collision free. If the robot used at this stage is not a disk, then collision checking is done by putting the robot at that midpoint and orienting it along the edge.

2.2 Approximate Roadmap

We next construct an approximate roadmap from the Cntl-Rdmp. Intuitively, we will use the Cntl-Rdmp to help us compose (rather than randomly generate) nodes in the car-like robot's $R^2 \times [0, 2\pi)$ C-space. The positional coordinates of the nodes are the midpoints on the edges of the Cntl-Rdmp, and their orientations are directed along the edges, so as to align them with the roadway. If the validity of these nodes was not already checked during construction of the Cntl-Rdmp, they will be checked now before adding them to the approximate roadmap.

In node connection, we connect nodes corresponding to adjacent edges in the Cntl-Rdmp. For each such connection, the two nodes are connected via a simple arc and a line segment if needed. For example (see Figure 2), consider two nodes $(\mathbf{r_1}, \theta_1)$ and $(\mathbf{r_2}, \theta_2)$, and let their common endpoint in the Cntl-Rdmp be $\mathbf{r_3}$, where $\mathbf{r_i} = (x_i, y_i)$. Also, let $a = \|\mathbf{r_1} - \mathbf{r_3}\|$ and $b = \|\mathbf{r_2} - \mathbf{r_3}\|$. Denote the angle between the two edges as α. Then the curvature for the arc is $\kappa = \cot(\alpha/2)/\min(a, b)$, and the length of path is $l = 1/\kappa * \alpha + |a - b|$. The edge is added to roadmap, without collision checking, if its curvature is smaller than some threshold κ_{max}. This coarse curvature test filters useless nodes and edges from the roadmap.

2.3 Querying – an adaptive approach

The roadmap can then be used to answer different queries. Notice that so far we have not mentioned the minimum turning radius of a specific car-like robot: the roadmap construction is not dependent on it.

Now assume we want to find a path from a start point to a goal point for a car-like robot which has minimum turning radius r_{min}. To do this, we first connect the start and the goal to the roadmap, and then remove all edges from the roadmap with curvature larger than $1/r_{min}$. After that, we use Dijkstra's algorithm to find the shortest path between the start and the goal. The path will consist of a sequence of nodes from the roadmap. Note that such a path could possibly contain several cusps, which indicate transitions between forward and backward movements of the car. From the perspective of the control roadmap, the path is determined by a sequence of control points. After we have a path, we must validate it with finer resolution collision tests since the validity of the edges has not been verified before. If any node or edge on the path fails to meet the query requirements, it is removed from the roadmap, a new shortest path is found in the refined roadmap, and the process iterates until a valid path is found or failure is reported.

Here, we can see how the same roadmap could be pruned to meet the planning requirements of cars with different turning radii, or indeed, any other requirement, such as a pre-set minimum clearance. For example, if we subsequently want to plan a path for a robot with an even larger turning radius, we can simply prune more edges from the previously pruned roadmap until a path satisfying the curvature requirement is obtained, or failure is reported.

3 Path Optimization: curvature properties

The path created above is made up of arcs and line segments. An unpleasant feature of this is that since the curvature is not continuous along the path, the robot has to make a complete stop each time it meets a curvature discontinuity. Since each roadmap node along the path corresponds to an edge in the control roadmap, it is a simple matter to retrieve all the control points along the path.

A natural question that arises is whether it is possible to find a better path, with fewer discontinuities in the curvature, using the guidance of these control points (or polygon). More precisely, we first partition the path at any cusps, into forward and reverse segments, and process each segment separately. Actually, it is convenient to reverse the statement and ask a more general question: given this control polygon, is it possible to get a collision free path with better curvature (by better, we mean bigger and/or more continuous curvature)?

In the following, we will show that for a given control polygon, arcs always perform better than quadratic B-splines in terms of obtaining smaller maximum curvature. The other option is cubic B-splines, which naturally provide continuous curvature if the path happens also to be collision free. We will show a heuristic that can be used to find better knot sequences that minimize the maximum curvature along the path.

3.1 Arcs and Quadratic B-splines

Compared with arcs and line segments, quadratic B-splines look smoother. However, arcs are better at minimizing the maximum curvature along a curved segment controlled by three control points (see Figure 2).

Consider the integration of the y component along both curves. We have $\int 1/\kappa * \sin(\theta) d\theta = Y = c_1$, constant c_1, for both curves. Since both integrations go through the same range of θ, it is impossible for the curvature of the quadratic B-spline curve to always be smaller than that of the arc, since otherwise its integration along the path would be larger than Y, the constant.

An alternative, more intuitive argument is to notice that if the spline curve were to always have a smaller curvature along its path, then we could draw a bigger arc inside it and still be tangent to the control polygon, which is impossible (a contradiction).

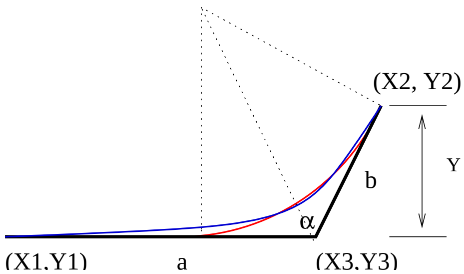


Figure 2: An arc and a quadratic B-spline in a control polygon.

3.2 Arcs and Cubic B-splines

Since we would like to have continuous curvature along the path and still follow the guidance of the control polygon, it is natural to consider cubic B-splines.

Since at this point we have already found a path, we can always return to the arc-line path if the cubic B-spline is in collision. So, let us put the collision issue aside, and ask what is the best knot sequence to minimize the maximum curvature along the path? Here, we propose a heuristic method which is based on the following observation. The cubic B-spline curve consists of a set of C^2 continuous Bezier curves. For each curve segment, we have $\int \kappa ds = \Delta\theta$, i.e., the integration of the curvature is the difference between the directions at the end points, which can be estimated as a constant. Since the curve length s is directly related to $\Delta u = u_{i+1} - u_i$, increasing the knot range Δu will lower the curvature along that curve segment.

The heuristic works as follows: starting with chord-length or uniform knot sequences, calculate the maximum curvature along each Bezier curve segment, and denote them as $C^1_{max}, C^2_{max}, \ldots, C^i_{max}, \ldots$. Denote also the knot range for each curve segment as $\Delta u_1, \Delta u_2, \ldots, \Delta u_i, \ldots$. Then, in the next iteration, reassign the knot sequence to ensure a new ratio among the Δu. The new ratio is $C^1_{max} * \Delta u_1 : C^2_{max} * \Delta u_2 : \ldots : C^i_{max} * \Delta u_i : \ldots$. Continue this process until the maximum curvature reaches a minimum (most likely a local minimum) or a certain iteration limit is reached.

Figure 3 shows one example, the result of a curve and its curvature before (uniform knot sequences, dashed line) and after running the heuristic (the solid line). One can that the maximum curvature along the curve becomes smaller after running the heuristic.

4 Implementation and Results

In this section, we show some results obtained using our C-PRM-based planning approach for nonholonomic car-like robots in several environments (see Figures 4–7). A summary of our results on these experiments is shown in Table 1. It can be seen that in all cases, the planner is able to find a fairly good solution in a short time.

All experiments were performed on a Pentium III 550 MHz PC using our C++ OBPRM library [1], which includes

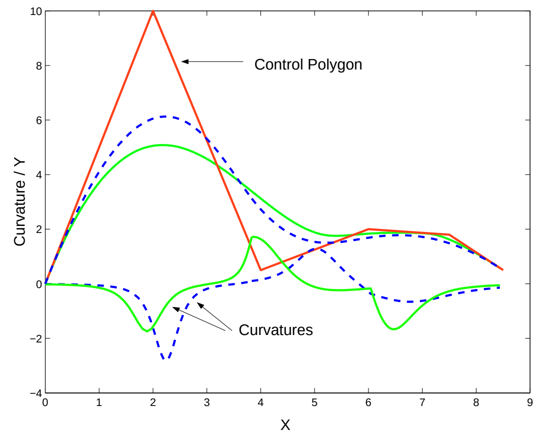


Figure 3: The running results of the heuristic methods.

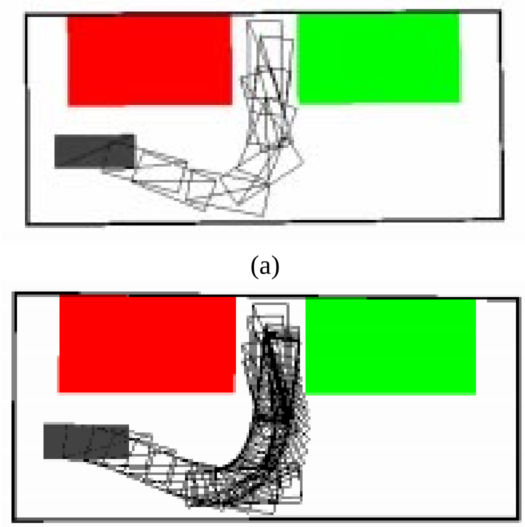
implementations of many PRM variants. In addition to traditional randomized PRM for node generation, some control roadmap nodes were generated using a variant of the medial axis PRM, or MAPRM [27, 28], which attempts to place nodes on the medial axis of the free C-space.

In all experiments described below, we use the same 'car'. Its size is 12 by 5 (environment units). The distance between the front and rear wheels is 8 units. The car can drive both forward and backward. The size of the environment (the bounding box) is 72 by 32 units. All curvature requirements are set as 6 units unless otherwise specified.

Running time				
Env.	Roadmap Construction #Node	#Edge	time[sec]	Query time[sec]
Scene 1	814	3249	3.26	5.81
Scene 2	3196	6316	15.82	6.52
Scene 3	4173	8805	32.49	9.94
Scene 4	1194	1991	2.50	4.43

Table 1: Roadmap statistics (number of nodes and edges) and running times for constructing and querying the roadmap.

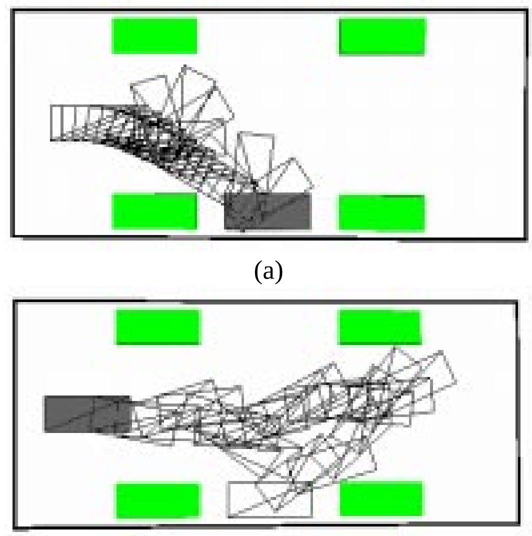
Scene I – Head-In Parking. In Figure 4, we show a head-in parking scenario. As seen in Figure 4(a), the path found by the planner looks quite realistic. To make a smooth turn, the car first turned a bit to the right, and then turned back to the left to the parking place, as people normally do. It took 3.26 seconds to construct the roadmap and 5.81 seconds for the query to find a path. It takes more time during the query since the roadmap is only approximately validated, and so this validation must be performed at the query time. As described earlier, the user can specify turning radius and clearance requirements. In this case, the bound on the turning radius was set to six environment units. Figure 4(b) shows the path from part (a) after it has been smoothed using a cubic B-spline (see Section 3.2). Only part of the path is smoothed in this case so as not to violate the turning radius requirements.



(a)

(b)

Figure 4: Scene I – Head-in parking: (a) a solution path, and (b) a new path after the old one has been partly smoothed using cubic b-spline.



(a)

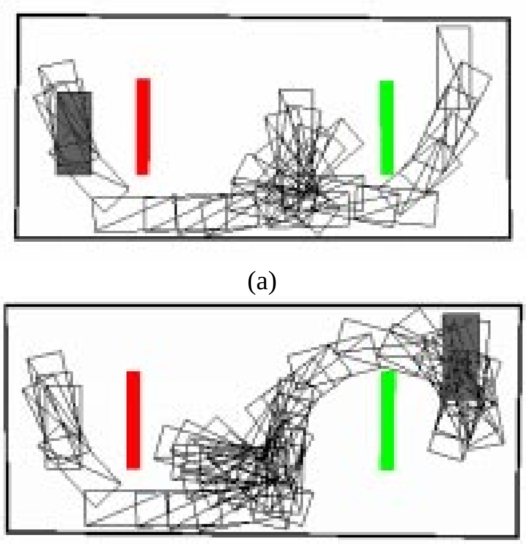
(b)

Figure 5: Scene II – Parallel Parking. Solution paths with (a) an unrealistic turning radius, and (b) a more realistic turning radius. In both cases, the same roadmap was used. The different turning radii were specified at query time.

Scene II – Parallel Parking. In Figure 5, we show a parallel parking scenario. It is interesting to note that under different turning radii requirements, the planner found different paths – starting from the same roadmap. As shown is Figure 5(a), with a smaller turning radius requirement the planner selected the shortest path and drove the car essentially straight into the spot. As the turning radius requirement approaches that of an actual car, the path looks more natural, see Figure 5(b). This implies that PRM-based path planning for car-like robots is plausible, and also suggests it is suitable for use in simulations, such as cars moving in virtual scenes and other animations.

Scene III – Driving around Obstacles. Figure 6 shows a scenario where the robot starts from the top-right corner and its goal is the center of the left corridor. Here we show the different paths the planner selects when the relative cost of forward and backward motion is changed. In this scenario, most human drivers would attempt to follow a path that looks like an ”S”. However, when both forward and backward driving are given equal weight, the planner finds the shortest (in terms of length) path, which involves a significant portion of backward movement (see in Figure 6(a)). In this case, the shortest route for the car is to back out from right corridor, turn around in the open space, drive into the left corridor, and continue until it reaches the goal.

However, one can encode preferences in the roadmap by weighting the edges. For example, in this situation, one might put put more weight (or a penalty) on backward driving by, e.g., increasing the length of backward edges by a factor of c, for some constant c. The new result, obtained from the same roadmap, with the new weights processed in the query phase, is shown is Figure 6(b). It can be seen that it does follow an ”S” shape, as desired.



(a)

(b)

Figure 6: Scene III – Driving around Obstacles. In (a), the shortest path requires a significant backwards segment, but in (b), an alternative, more ‘natural’, path is obtained when backward driving is penalized by a factor of 10.

Scene IV – Navigating around Many Obstacles. Now we test the algorithm in a scene with many (19) randomly placed triangles. The control map and the real roadmap for this scene are shown in Figure 1. From the roadmaps, it can clearly be seen that the method is able to construct a fairly good roadmap. It seems the planner is able to lay down "roads" in most of the drivable space, even though the scene is quite irregular. A solution path moving from the lower-left to the upper-right corner is found in a few seconds, as shown is Figure 7.

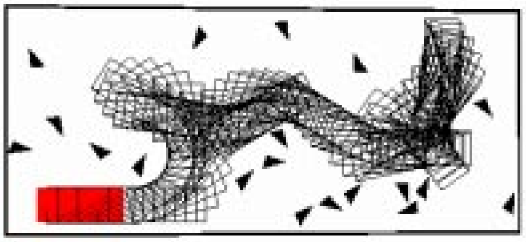


Figure 7: Scene IV – Navigating around many Obstacles. Navigating in a complex scene with 19 randomly placed triangles.

5 Conclusion

In this paper, we present a new PRM-based approach for car-like robot motion planning which is based on the Customizable PRM (C-PRM). We describe how to use a so-called 'control roadmap' to effectively generate good configurations in the robot's C-space and then swiftly connect them into the final roadmap. A major strength of the method is that the *same* roadmap can be customized to do motion planning for car-like robots with different turning radii. A heuristic is also proposed for smoothing paths using cubic B-splines.

Acknowledgements

Song would like to thank Prof. John Keyser for his excellent class on Geometric Modeling, in which the preliminary version of the work was developed. We thank the anonymous reviewers for their constructive suggestions and comments. We also would like to thank the motion planning group at Texas A&M for providing much support.

References

[1] 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)*, pages 155–168, 1998.

[2] E. Anshelevich, S. Owens, F. Lamiraux, and L. Kavraki. Deformable volumes in path planning applications. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000.

[3] M.S. Apaydin, A.P. Singh, D.L. Brutlag, and J.-C. Latombe. Capturing molecular energy landscapes with probabilistic conformational roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 932–939, 2001.

[4] O. B. Bayazit, G. Song, and N. M. Amato. Enhancing randomized motion planners: Exploring with haptic hints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 529–536, 2000.

[5] O. B. Bayazit, G. Song, and N. M. Amato. Ligand binding with OBPRM and haptic user input: Enhancing automatic motion planning with virtual touch. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 954–959, 2001. This work was also presented as a poster at RECOMB'01.

[6] A. Bemporad, A. De Luca, and G. Oriolo. Local incremental planning for a car-like robot navigating among obstacles. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1205–1211, 1996.

[7] R. Bohlin and L. E. Kavraki. Path planning using lazy prm. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 521–528, 2000.

[8] H. Chang and T. Y. Li. Assembly maintainability study with motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1012–1019, 1995.

[9] J.-P. Laumond (*Editor*). *Robot Motion Planning and Control*. Springer, London, 1998.

[10] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000.

[11] L. Kavraki, F. Lamiraux, and C. Holleman. Towards planning for elastic objects. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998.

[12] 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.*, 12:566–580, 1996.

[13] M. Khatib, H. Jaouni, R. Chatila, and J.P. Laumond. Dynamic path modification for car-like nonholonomic mobile robots. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2920–2925, 1997.

[14] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[15] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.

[16] S.M. LaValle, J.H. Yakey, and L.E. Kavraki. A probabilistic roadmap approach for systems with closed kinematic chains. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999.

[17] C. L. Nielsen and L. E. Kavraki. A two level fuzzy prm for manipulation planning. Technical Report TR2000-365, Computer Science, Rice University, Houston, TX, 2000.

[18] J.A. Reeds and R.A. Shepp. Optimal paths for a car that goes both forward and backward. *Pacific Journal of Mathematics*, 145(2):367–393, 1991.

[19] J. Reif. Complexity of the piano mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, 1979.

[20] A.P. Singh, J.C. Latombe, and D.L. Brutlag. A motion planning approach to flexible ligand binding. In *7th Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 252–261, 1999.

[21] G. Song, , S.L. Miller, and N. M. Amato. Customizing PRM roadmaps at query time. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1500–1505, 2001.

[22] G. Song and N. M. Amato. A motion planning approach to folding: From paper craft to protein folding. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 948–953, 2001.

[23] G. Song and N. M. Amato. Using motion planning to study protein folding pathways. In *Proc. Int. Conf. Comput. Molecular Biology (RECOMB)*, pages 287–296, 2001.

[24] P. Souères and J.-P. Laumond. Shortest paths synthesis for a car-like robot. *IEEE Transactions on Automatic Control*, 41:672–688, 1996.

[25] S. Sundaram, I. Remmler, and N.M. Amato. Disassembly sequencing using a motion planning approach. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1475–1480, 2001.

[26] P. Švestka. A probabilistic approach to motion planning for car-like robots. Technical Report RUU-CS-93-18, Dept. of Computer Science, Utrecht University, April 1993.

[27] 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)*, pages 1024–1031, 1999.

[28] 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)*, pages 173–180, 1999.