

Incremental Map Generation (IMG)

Dawen Xie Shawna Thomas Jyh-Ming Lien Nancy M. Amato
dawenx@cs.tamu.edu sthomas@cs.tamu.edu neilien@cs.tamu.edu amato@cs.tamu.edu

Technical Report TR05-007
Parasol Lab.
Department of Computer Science
Texas A&M University

September 21, 2005

Abstract

Automatic motion planning has applications ranging from traditional robotics to computer-aided design to computational biology and chemistry. Randomized planners, such as probabilistic roadmap methods (PRMs), have been highly successful in solving these high degree of freedom problems. However, the traditional PRM framework fails to address several practical issues. One of the most important issues is the difficulty of deciding what size roadmap is required to solve a given problem efficiently. PRMs do not provide an automated way to determine appropriate roadmap size. In this paper, we propose a new PRM-based framework called Incremental Map Generation (IMG) to address this problem. Our strategy is to break the map generation into independent processes. Each process generates an independent roadmap component. IMG proceeds by adding independent roadmap components to an existing roadmap until some user defined criteria are satisfied. In addition to addressing the roadmap size problem, this framework supports roadmap reproducibility in that any of the roadmap increments can be reproduced by using the same set of seeds. Finally, these independent processes are natural for parallelization.

1 Introduction

Automatic motion planning has applications in many areas such as robotics [26], computer animation, computer-aided design/virtual prototyping, and computational biology and chemistry. Although many deterministic motion planning methods have been proposed, most are not used in practice because they are computationally infeasible except for some restricted cases, e.g., when the robot has few degrees of freedom (dof) [22, 26]. Indeed, there is strong evidence that any complete planner (one that is guaranteed to find a solution or determine that none exists) requires time exponential in the number of dof of the robot [34].

For this reason, attention has focused on randomized approaches that sample and connect points in the movable object’s configuration space (C-space). Such methods include *probabilistic roadmap methods* (PRMs) [24], along with its different extensions and variants [1, 10, 40], for multiple query processing and tree-based methods [21, 27, 31] for single query processing. PRMs have been highly successful in solving challenging problems with many dof that were previously unsolvable and have become the method of choice for a wide range of applications such as robotics [26], group behaviors [6, 28], deformable objects [4, 7], computer-aided design/virtual prototyping [8, 15], ligand docking [9, 35], and protein folding [5, 36].

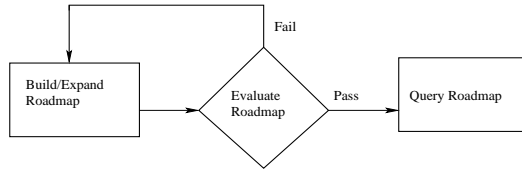


Figure 1: Flow diagram for Incremental Map Generation (IMG).

However, the traditional PRM framework does not address several practical issues. One of the most important issues is the difficulty of deciding beforehand what size roadmap is required to capture the connectivity of the free C-space for a given problem. The PRM framework does not provide an automated way to build a roadmap that is large enough to adequately capture the connectivity but small enough to be efficient. In practice, the user selects a roadmap size they believe is appropriate. If the roadmap is too small, it may falsely report that no path exists. The user may expand the existing roadmap as in [24], but results are not reproducible unless there is careful tracking of the random number generator seeds. If the user is concerned about reproducibility, they must build an entirely new roadmap of larger size. For some applications such as protein folding where roadmaps can take days or weeks to build [37, 38], this is impractical. On the other hand, if the roadmap is too big, valuable computation time is wasted in building the roadmap and more time will be required to process a query.

1.1 Our Contribution

In this paper we propose a PRM-based framework called Incremental Map Generation (IMG) to automatically build a roadmap of an appropriate size for a given problem. This is implemented by iteratively building the roadmap until it satisfies a set of evaluation criteria (see Figure 1). This differs from the traditional two-phase PRM method [24] in two aspects. First, we include a new phase called “evaluate roadmap” which tests if the roadmap passes a set of evaluation criteria. If so, preprocessing ends and query processing begins. If not, preprocessing continues by expanding the roadmap by a fixed size. This phase provides some measure of roadmap quality. Second, we partition roadmap construction into several *independent sets* where we perform sampling and connection within and among each set. Each computation set obtains a unique random seed and uses this seed to generate its own roadmap. Note that IMG is *not* a new sampling method but is a general strategy that can be applied to any sampling-based planner.

The most important feature of IMG is that it provides a framework to automatically determine the appropriate roadmap size for a given problem. The framework can accept a broad range of stopping or evaluation criteria which can be customized for particular applications or user preferences.

In addition to automating the roadmap generation process, independent roadmap computation provides several benefits, including:

- providing roadmap reproducibility,

- support for different roadmap generation strategies, and
- providing natural functionality for parallel implementations.

Roadmap reproducibility. Our framework can support roadmap reproducibility by using a deterministic process to set the seed for each component and to record the associated roadmap generation strategies and the creation order of each independent component. Note that this reproducibility remains valid even after more independent components are added to an existing roadmap.

Planning strategies. New roadmap generation strategies can be introduced into the roadmap construction process by adding another independent computation. For example, more powerful sampling strategies can be dynamically added to the map generation as more difficult areas, such as narrow passages, are identified.

Parallelization. Another benefit of our framework is that these independent computations can be easily parallelized. For example, while parallel implementations of randomized motion planners [3, 12–14, 30, 33, 37, 38] do not provide solution consistency when varying the number of processors, our framework can provide a consistent method for seeding random number generators, eliminating this processor count dependence.

The paper is outlined as follows: in Section 2 we provide a brief overview of related work in the area. We present our IMG framework in Section 3. In Section 4 we show experimental results for several applications and discuss our findings. Finally, in Section 5 we address future work.

2 Related work

The general PRM methodology [24] consists of a preprocessing phase and a query phase. Preprocessing, which is done once for a given environment, first samples points ‘randomly’ from the robot’s C-space, retaining those that satisfy certain feasibility requirements. Then, these points are connected to form a graph, or roadmap, that represents the free C-space. The query phase then connects any given start and goal configuration to the roadmap and returns a path if one exists in the roadmap.

The probability of failing to find a path from a probabilistic roadmap given one exists decreases exponentially as the number of samples in the roadmap increases [23]. However, it is difficult to decide beforehand what size roadmap is required in practice. We propose a framework that automatically decides how large a roadmap to build based on a set of user defined evaluation criteria. One difficulty is in deciding appropriate evaluation criteria. In [18], the authors predefine a relevant query in each test scene and continue building the roadmap until the query configurations are in the same connected component. In [19], coverage and maximal connectivity were used as an analysis tool to gain insight in sampling based methods. In particular, coverage ensures that every query can be directly connected to the roadmap graph. If there exists a path in free C-Space between two query configurations, maximal connectivity ensures that a path between them can be found in the roadmap graph [19]. The authors discretized C-space and determined for various techniques how long it takes before the free C-space has been adequately covered and connected.

3 Incremental Map Generation (IMG)

We propose a new PRM-based framework called Incremental Map Generation (IMG) in which we iteratively build a roadmap until it satisfies a set of evaluation criteria. This framework automates the roadmap construction process in a systematic way that automatically decides how large a roadmap to build based on user selected evaluation criteria. Figure 1 illustrates our overall approach and the algorithm is given in Algorithm 3.

3.1 Incremental roadmap construction

To build the map incrementally, we divide roadmap construction into equal independent “sets” of a user specified size. In order to ensure the independence of each set, we reseed the random number generator (RNG) for each set based on the *base seed* of the program (e.g., the time execution starts), the type of node generation method, and the number of sets completed by that node generation method so far. This provides several advantages:

- calculating the seed in a deterministic way based on a (possibly random) base seed supports reproducibility,

Algorithm 3.1 Incremental Map Generation.

Input. An existing roadmap R , a roadmap evaluator E , the size of a node set, n , and the number of sets to construct before evaluating, k .

Output. A roadmap R that meets all the criteria indicated by E .

```
1: repeat
2:   for  $i \in \{0, 1, \dots, k\}$  do
3:     Let  $s$  be the new seed for node set  $i$ .
4:     Seed the RNG with  $s$ .
5:     Construct a new roadmap  $R_i$  with  $n$  nodes.
6:     Let  $R' = R \cup R_i$ .
7:     Add additional connections between nodes in  $R_i$  and nodes in  $R$ .
8:     Set  $R = R'$ .
9:   end for
10: until  $R$  meets criteria in  $E$ 
```

- adding independent roadmap components at run-time can allow map generation strategies to adapt as areas of C-space are characterized, and
- generating each set independently facilitates parallelization.

We generate an equal number of nodes in each set. For each set of nodes, we perform connection among them with a user specified set of connection strategies. After each set is constructed, we merge this new set with the existing roadmap by adding additional connections between the new nodes and the existing nodes with a (possibly different) user specified set of connection strategies. We repeat this process k times and then evaluate the roadmap, where k is specified by the user. We continue constructing the roadmap in this way until it meets all the evaluation criteria.

3.2 Roadmap Evaluation

The IMG framework continues to construct a roadmap until it meets a set of evaluation criteria (see Figure 1). These criteria can range from as simple as solving a user specified query to more complicated such as requiring the maximum network flow to exceed a threshold between two configurations. In this paper, we give several examples of roadmap evaluators and study their performance in various situations.

Query Evaluation. This evaluator simply determines whether a roadmap can solve a user specified query. It attempts to connect the start and goal to the roadmap, and returns successful if they are connected to the same connected component. This type of evaluator is useful when the user wants to solve a particular test problem.

Coverage Evaluation. This evaluator examines how well the roadmap covers the C-space. It attempts to connect c nodes, either randomly generated or user specified, to the roadmap. It returns success if it can connect at least $x\%$ of them, where x is a user specified threshold.

Connectivity Evaluation. This evaluator is a more generalized version of the query evaluator. It attempts to connect c nodes, either randomly generated or user specified, to the roadmap. It returns success if $x\%$ of all possible pairs in c are connected to the same connected component, where x is a user specified threshold. This gives a measure of what amount of queries this roadmap can solve.

Roadmap Expansion Evaluation. This evaluator attempts to measure when the roadmap topography is no longer changing. It computes the inter-component distances and intra-component distances of the connected components of a given roadmap. These distances can be defined in several ways. For the results in this paper, we define them as follows. The inter-component distance between two connected components is defined as the distance between their centers of mass and a component's center of mass is simply the average of all nodes in the component. The intra-component distance of a connected component is defined as the distance between the two furthest configurations in the component. The evaluator returns success if both sets of distances stabilize.

Max-flow Evaluation. Some applications not only require a single path between two configurations but many paths between them. For example, motion planning has been recently applied to study problems in computational biology such as protein folding and protein structure transitions [5, 36]. To study how a protein changes from one target configuration to another, we can examine the probable paths between them in the roadmap. We can

define this as a maximum flow problem on a network. If a roadmap edge weight, $w(e)$, reflects the likelihood that the protein will move from one configuration to the next, then we can define edge capacity $c(e)$ as $1/w(e)$. The evaluator returns success if the max-flow between the two configurations is above some threshold f .

4 Experimental Results

In this section, we show how IMG performs in practice.

4.1 Experimental Setup

For all experiments we use PQP [25] for collision detection calculations. Two types of local planners, straight-line and rotate at 0.5 [2], are used to connect sampled configurations. For nodes within an independent computation set, we only attempt to connect the 10 nearest neighbors, using euclidean distance in C-space. This results in several connected components that must be merged with the existing roadmap. For each new connected component, we examine the 3 nearest components in the existing roadmap based on inter-component distance as described earlier. For each of these component pairs, we attempt to connect the 10 nearest pairs of nodes. All results were run on 700MHz Intel PIII Xeon processors.

Recall that IMG is *not* a new sampling method but is a general strategy that can be applied to any sampling-based planner. We investigate how IMG automatically builds roadmaps of an appropriate size using different evaluation criteria. In all experiments, we used the following sampling methods:

- *Uniform random sampling*: samples are created by picking random values for all degrees of freedom.
- *Gaussian-biased random sampling* [11]: first 2 samples are created, one uniformly at random and one a distance d away, where d has a Gaussian distribution, until 1 sample is collision-free and the other is not; the collision-free sample is added to the roadmap.
- *Bridge-test random sampling* [20]: similar to Gaussian sampling, it takes two random samples, a distance d apart, where d has a Gaussian distribution, until both samples are in collision and their midpoint is not; the collision-free sample is added to the roadmap.
- *Obstacle-based sampling* (OBPRM) [1]: samples are generated approximately on C-obstacle surfaces by first generating a random colliding (reps., collision-free) sample and searching along a random direction until the sample becomes collision-free (resp., in collision).
- *Medial axis-based sampling* (MAPRM) [29, 40]: samples are generated approximately on the medial axis of the free C-space by first sampling at random, pushing colliding samples to C-free in the direction of smallest penetration and pushing collision-free samples away from the nearest C-obstacle until it is equally near 2 or more C-obstacles.

4.2 Automatically building roadmaps of appropriate size

We investigate the performance of the evaluators described in Section 3.2 in the four different environments in Figure 2, three rigid body problems and one articulated linkage with varying dof.

For coverage evaluation, we generate 100 samples uniformly at random and report success if 95% can be connected to the roadmap. For connectivity evaluation, we use the same samples as coverage evaluation and report success if 75% of the possible queries are solvable. To test one single evaluator, we use all 5 previously mentioned sampling strategies. We let each sampling strategy generate 30 roadmaps and the results in Table 1 are averaged over 150 roadmaps to get an adequate sample of the problem space.

To determine how well the final roadmaps reflect the connectivity of the free C-space, we generated an independent test set of 100 samples, 20 from each node generation method. We then compute what percentage of samples are connectible to the roadmap and what percentage of queries are solvable with the roadmap. These are indicted in Table 1 under the “Success Rate” columns.

From Table 1, we can see that roadmaps generated using coverage evaluation have success rate, thus quality, lower than the maps generated using predefined query and connectivity evaluations in all environments. In general,

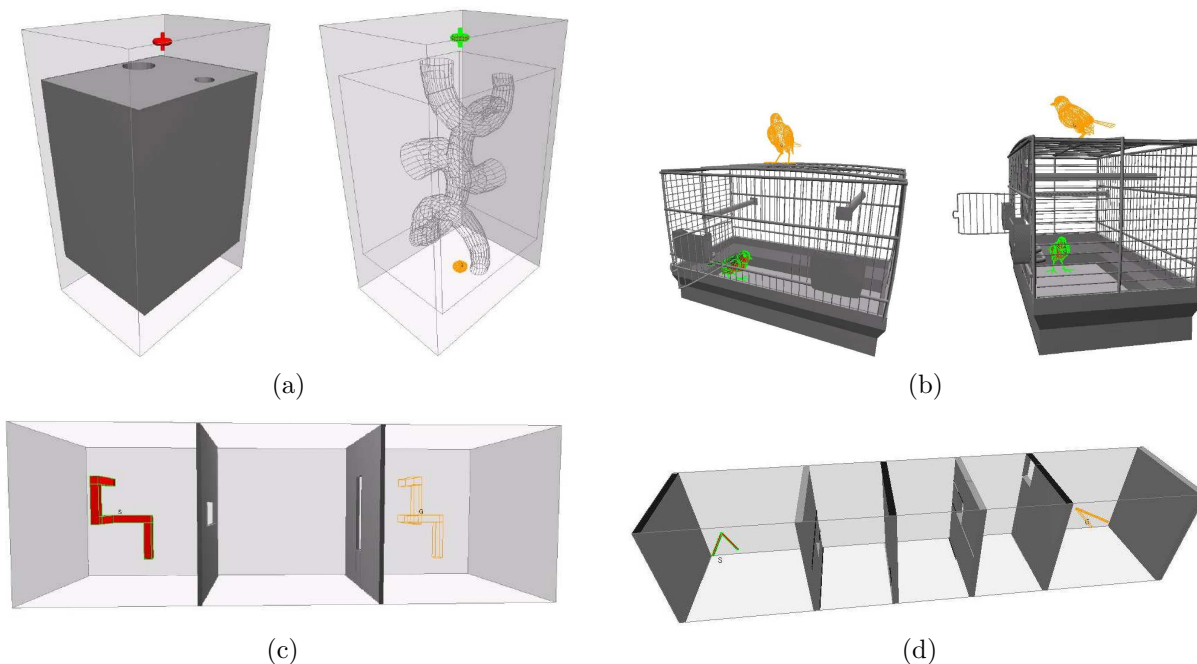


Figure 2: Problems studied. (a) Maze environment (solid/wire frame): rigid body robot must navigate the maze. (b) Cage environment (front/side): rigid body robot must navigate from inside the cage to outside the cage. (c) Hook environment: rigid body must rotate to move from one side of the wall to the other. (d) Walls environment: articulated linkage must travel through each narrow hole to get from one end of the corridor to the other.

connectivity evaluation can produce similar or even better roadmaps with fewer nodes than predefined query, except in the cage environment. This is because many uniformly sampled configurations used for connectivity evaluation are generated outside of the cage but many of the configurations used for computing success rate are close to the surface of the cage. We expect improvement from connectivity evaluation when a mixture of configurations generated using different sampling strategies are used.

As discussed in Section 3.2, we can apply our framework to study computational biology problems such as protein folding and protein structure transitions. Here, we incrementally build a roadmap until the maximum flow between the two configurations of interest is above a threshold. We applied this approach to study calmodulin, a signaling protein that binds to Ca^{2+} to regulate several processes in the cell [16, 17, 39]. When calmodulin binds to Ca^{2+} , it undergoes a large-scale rearrangement [32] shown in Figure 3.



Figure 3: Rearrangement of Calmodulin: (a) calcium-free state (1CFD) to (b) bound state (1CLL).

Using IMG, we were able to build a roadmap capturing the C-space around both target configurations as well as the transition between them in 2 weeks of computation time. The traditional method would take at least 4 weeks of computation time to produce the same size roadmap. Thus, for applications that take significant computing resources, it is imperative that we reuse the previous computation and build the roadmap incrementally. IMG

Table 1: Comparison of different roadmap evaluation methods. Results for each evaluator are averaged over 150 roadmaps generated using uniform, gauss, bridge, obprm and maprm sampling strategies.

Maze				Cage			
Evaluation Method	Size	Success Rate		Evaluation Method	Size	Success Rate	
		Conn.	Solved			Conn.	Solved
Predefined Query	2600.0	98.7%	89.7%	Predefined Query	3261.7	91.6%	81.4%
95% Coverage	250.0	96.7%	41.4%	95% Coverage	250.0	89.1%	47.3%
75% Connectivity	2971.7	99.4%	94.7%	75% Connectivity	403.3	90.4%	52.6%

Hook				Walls with 2 Links			
Evaluation Method	Size	Success Rate		Evaluation Method	Size	Success Rate	
		Conn.	Solved			Conn.	Solved
Predefined Query	18368.0	90.4%	69.7%	Predefined Query	14453.7	83.8%	63.7%
95% Coverage	250.0	87.4%	25.0%	95% Coverage	250.0	82.6%	16.0%
75% Connectivity	16717.2	90.9%	69.3%	75% Connectivity	12460.1	79.4%	62.5%

provides a framework to do so.

5 Conclusion

In this paper, we proposed a framework to automatically determine the appropriate roadmap size for a given motion planning problem. The framework can accept a broad range of evaluation criteria which can be customized for particular applications. In addition to addressing the roadmap size issue, our framework provides other benefits of supporting roadmap reproducibility and different roadmap generation strategies, and easy parallelization.

There are several areas we would like to investigate further. First, we would like to expand our list of node generation methods to include other types of random sampling and grid-based techniques. Second, the current bottleneck of our implementation is in merging the new independent computation with the existing roadmap. We use a naive approach here that only attempts connections between nearby components. We would like to further explore techniques for connecting more efficiently. Also, we examined how each evaluator performs individually. In the future, we would like to study combinations of evaluators and their performance in various types of problems. We would also like to investigate more sophisticated evaluation techniques such as one based on information gain. Such an evaluator would report that the roadmap is sufficient if the information gain from the previous set of samples is below a threshold.

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 *Robotics: The Algorithmic Perspective*, pages 155–168, Natick, MA, 1998. A.K. Peters. Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.
- [2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. Robot. Automat.*, 16(4):442–447, August 2000. Preliminary version appeared in ICRA 1998, pp. 630–637.
- [3] N. M. Amato and L. K. Dale. Probabilistic roadmap methods are embarrassingly parallel. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 688–694, 1999.
- [4] E. Anshelevich, S. Owens, F. Lamiroux, and L. Kavraki. Deformable volumes in path planning applications. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2290–2295, 2000.
- [5] M. Apaydin, A. Singh, D. 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.
- [6] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Better group behaviors in complex environments using global roadmaps. In *Artif. Life*, pages 362–370, Dec 2002.

- [7] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Probabilistic roadmap motion planning for deformable objects. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2126–2133, May 2002.
- [8] 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.
- [9] 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 2001*.
- [10] R. Bohlin and L. E. Kavraki. Path planning using Lazy PRM. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 521–528, 2000.
- [11] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1018–1023, 1999.
- [12] S. Carpin and E. Pagello. On parallel rrts for multi-robot systems. In *Proc. Italian Assoc. AI*, pages 834–841, 2002.
- [13] D. Challou, D. Boley, M. Gini, and V. Kumar. A parallel formulation of informed randomized search for robot motion planning problems. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 709–714, 1995.
- [14] D. J. Challou, M. Gini, and V. Kumar. Parallel search algorithms for robot motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 46–51, 1993.
- [15] H. Chang and T. Y. Li. Assembly maintainability study with motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1012–1019, 1995.
- [16] A. Crivici and M. Ikura. Molecular and structural basis of target recognition by calmodulin. *Annu. Rev. Biophys. Biomol. Struct.*, 24:85–116, 1995.
- [17] J. Evenas, A. Malmendal, and S. Forsen. Calcium. *Curr. Op. Chem. Biol.*, 2(2):293–302, 1998.
- [18] R. Geraerts and M. H. Overmars. Sampling techniques for probabilistic roadmap planners. In *Proc. Conf. on Intel. Auton. Syst. (IAS)*, pages 600–609, 2004.
- [19] R. Geraerts and M. H. Overmars. Reachability analysis of sampling based planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 406–412, 2005.
- [20] D. Hsu, T. Jiang, J. Reif, and Z. Sun. Bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4420–4426, 2003.
- [21] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2719–2726, 1997.
- [22] Y. K. Hwang and N. Ahuja. Gross motion planning – a survey. *ACM Computing Surveys*, 24(3):219–291, 1992.
- [23] L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *Proc. ACM Symp. Theory of Computing (STOC)*, pages 353–362, May 1995.
- [24] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.
- [25] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Distance queries with rectangular swept sphere volumes. In *Proc. of IEEE Int. Conference on Robotics and Automation*, 2000.
- [26] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.
- [27] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.
- [28] J.-M. Lien, O. B. Bayazit, R.-T. Sowell, S. Rodriguez, and N. M. Amato. Shepherding behaviors. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4159–4164, April 2004.
- [29] J.-M. Lien, S. L. Thomas, and N. M. Amato. A general framework for sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4439–4444, September 2003.
- [30] T. Lozano-Pérez and P. O’Donnell. Parallel robot motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1000–1007, 1991.
- [31] E. Mazer, J. M. Ahuactzin, and P. Bessiere. The Ariadne’s clew algorithm. In *Journal of Artificial Robotics Research (JAIR)*, volume 9, pages 295–316, 1998.
- [32] M. R. Nelson and W. J. Chazin. An interaction-based analysis of calcium-induced conformational changes in Ca^{2+} sensor proteins. *Protein Sci.*, 7:270–282, 1998.

- [33] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki. Sampling-based roadmap of trees for parallel motion planning. *IEEE Trans. Robot. Automat.*, 2005.
- [34] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, October 1979.
- [35] A. Singh, J. Latombe, and D. Brutlag. A motion planning approach to flexible ligand binding. In *7th Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 252–261, 1999.
- [36] 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.
- [37] S. Thomas and N. Amato. Parallel protein folding with stapl. In *Proc. 3rd IEEE International Workshop On High Performance Computational Biology (HiCOMB). Held with IPDPS.*, Santa Fe, NM, March 2004.
- [38] S. Thomas, G. Tanase, L. K. Dale, J. M. Moreira, L. Rauchwerger, and N. M. Amato. Parallel protein folding with stapl. *Concurrency and Computation: Practice and Experience*, 2005.
- [39] H. Vogel. Calmodulin: a versatile calcium mediator protein. *Biochem. Cell Biol.*, 72(9-10):357–376, 1994.
- [40] 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)*, volume 2, pages 1024–1031, 1999.