Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain *

Anna Yershova† Léonard Jaillet‡ Thierry Siméon‡ Steven M. LaValle†

†*Department of Computer Science*
University of Illinois
Urbana, IL 61801 USA
{yershova, lavalle}@uiuc.edu

‡*LAAS-CNRS*
7, Avenue du Colonel Roche
31077 Toulouse Cedex 04,France
{ljaillet, nic}@laas.fr

***Abstract*— Sampling-based planners have solved difficult problems in many applications of motion planning in recent years. In particular, techniques based on the Rapidly-exploring Random Trees (RRTs) have generated highly successful single-query planners. Even though RRTs work well on many problems, they have weaknesses which cause them to explore slowly when the sampling domain is not well adapted to the problem.**

In this paper we characterize these issues and propose a general framework for minimizing their effect. We develop and implement a simple new planner which shows significant improvement over existing RRT-based planners. In the worst cases, the performance appears to be only slightly worse in comparison to the original RRT, and for many problems it performs orders of magnitude better.

***Index Terms*— Motion Planning, Voronoi Bias, RRTs**

I. Introduction

Motion planning has many applications in such areas as robotics, manufacturing, pharmaceutical drug design, computational biology and computer graphics. The computational intractability of the general motion planning problem has led to the development of the sampling-based algorithms, which have been successfully used in practice over the last decade. Unfortunately, there is no planning algorithm that efficiently solves all instances of the motion planning problem. Each planner has a set of applicable problems, on part of which it performs well, and on another part it may perform poorly. Designing a single algorithm that reliably performs on a large set of problems is important to the motion planning area.

Current sampling-based algorithms can be divided into two sets of approaches: multiple-query and single-query methods. The primary philosophy behind the multiple-query methods is that substantial precomputational time may be taken so that multiple queries for the same environment can be answered quickly. The Probabilistic Roadmap (PRM) planner [19] is an example of such method. More recent roadmap methods are based on importance sampling. They

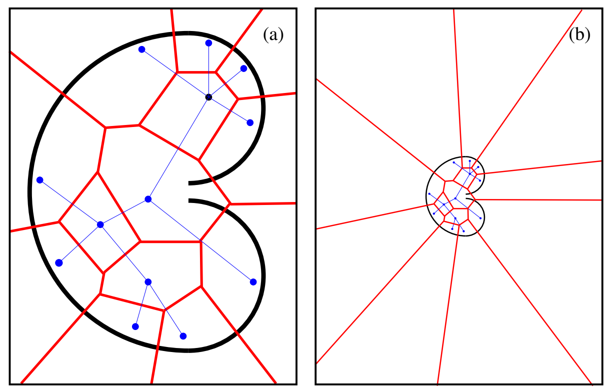


Fig. 1. A bug trap problem in high dimensions can be a challenging problem for RRT-based planner (a). It becomes much more challenging when a region for sampling is enlarged (b). The tree constructed by the RRT planner is shown in blue (black nodes and edges) and the Voronoi regions associated with the nodes of the tree are shown in red (gray edges).

concentrate samples in a nonuniform way, for example, along the C-space boundaries [1], [4], or the medial axis [14], [28], [35]. These and other methods are primarily designed for solving problems with narrow corridors [12], [15]. The visibility approach taken in [31] leads to another nonuniform way of sampling over the configuration space. PRMs can also be extended to problems of motion planning for closed chains [9], [13], [36], multiple robots [34], and nonholonomic robots [30], although there are difficulties with handling complicated differential constraints with PRM methods.

Multiple-query methods may take considerable precomputation time, thus different approaches were developed for solving single-query problems [3], [16], [23], [27], [29]. Rapidly-exploring Random Trees (RRTs) were primarily designed for targeting single-query holonomic problems and problems with differential constraints [21], [23], [25]. The performance success of the RRTs on many motion planning problems has led to broad extensions and applications of this approach. For example, problems with complicated geometries are handled with the RRT-based planners in [7],

*This work is partially supported by NSF CAREER Award IRI-9875304, NSF ANI-0208891, NSF IIS-0118146, the UIUC-CNRS grant, and the European project MOVIE, IST-20001-39250.

[10]. Extensions of RRTs for manipulation problems and motions of closed articulated chains are developed in [8], [17], [18], [33]. Adapted versions of RRTs for kinodynamic and nonholonomic planning also exist [5], [6], [11], [20], [22], [25]. In the place of random sampling used in these, deterministic, resolution-complete alternatives of RRTs have been proposed in [24], [26].

Even though RRTs work well in many applications, they have several weaknesses, which cause them to perform poorly in some cases. This is the reason why adaptations and extensions of RRTs are proposed to handle different classes of problems. In this paper, we characterize these issues and propose a general framework for minimizing the effect of some of these weaknesses. We have developed and implemented a simple new planner that shows significant improvement over existing RRT-based planners, in some cases by several orders of magnitude. The resulting planner treats some of the pathological cases of the original RRT, and at the same time is adapted to solve more classes of motion planning problems. By taking into account the obstacles of the configuration space into the Voronoi bias, the planner solves problems that have complicated geometries more efficiently. Although the idea is general enough and should be applicable to other constrained motion planning problems (e.g. planning for closed chains, nonholonomic planning), in this work we concentrated only on holonomic problems.

In the next section we re-examine the Voronoi biased exploration strategy of the RRTs and illustrate the performance of the RRT algorithm on one challenging example. In the end of section II, we formulate the problem of controlling Voronoi bias for more efficient exploration of the configuration space. We propose our solution to this problem in Section III and the experimental evaluations in Section IV.

II. Voronoi Bias Exploration in RRTs

RRTs were originally introduced in [23]. Starting at a given initial configuration, RRTs incrementally search the configuration space for a path connecting the initial and the goal configurations. At each iteration a new configuration is sampled and the extension from the nearest node in the tree toward this sample is attempted. If the extension succeeds, a new node in the tree is created.

There are several planners that exploit the exploration properties of the basic RRTs, such as the RRT-CONNECT planner (pseudocode shown in Figure 2). Bidirectional versions of RRTs exist (bi-RRTs), which alternate execution of the basic algorithm for two trees growing from the initial and the goal configurations, and put some additional bounds on the sizes of each of the trees (bidirectional balanced RRTs).

RRT exploration is determined by the Voronoi diagram of the nodes in the tree. The probability that a node will be chosen for an extension is proportional to the volume of its Voronoi region. Therefore, the RRT tends to rapidly grow in the unexplored regions of the configuration space.

BUILD_RRT(q_{init})
  $\mathcal{T}.init(q_{init})$;
  for $k = 1$ **to** K **do**
    $q_{rand} \leftarrow$ RANDOM_CONFIG();
    $q_{near} \leftarrow$ NEAREST_NEIGHBOR($q_{rand}, \mathcal{T}$);
    if CONNECT($\mathcal{T}, q_{rand}, q_{near}, q_{new}$)
      $\mathcal{T}$.add_vertex(q_{new});
      $\mathcal{T}$.add_edge(q_{near}, q_{new});
  Return $\mathcal{T}$;

Fig. 2. The RRT-CONNECT construction algorithm.

A. Motivating Example

Consider a problem shown in Figure 1(a). The task is to move the robot outside of the *bug trap*[1]. Since the size of the free space inside the trap is considerably larger than the narrow opening, in high dimensions it can be a very challenging problem for any motion planner. Now consider the Voronoi regions of the set of sample points built by the RRT planner inside the trap. There is a considerable bias toward the points near the obstacles. That is, most of the points on the boundary of the trap will have higher probability of being selected for extension than the points near the opening accordingly to the sizes of the corresponding Voronoi regions.

The problem becomes even more challenging if the sampling region is enlarged. It may become so difficult that a regular RRT will not be able to solve the 2-dimensional problem shown in 1(b) in any reasonable time. This is explained by an even larger bias toward the points near the obstacles. In fact, the Voronoi regions of these points grow with the size of the environment. Meanwhile, the tree in the middle of the bug trap does not grow at all.

The obvious solution to this problem would be to limit the sampling region to fit the bug trap. However, while this would help in this easy case, the approach would not be general enough to deal with other motion planning problems.

In what follows we distinguish different types of nodes in the tree. We call *frontier* nodes the vertices in the tree that have their Voronoi regions growing together with the size of the environment. The *boundary nodes* are those that lie in some proximity to the obstacles. It is important to note that the frontier vertices provide especially strong bias toward the unexplored portions of the configuration space. In many cases this helps the tree to rapidly grow. However, this may cause a slow-down in the performance when a frontier point is also a boundary point. For example, in the Figures 1(a) and (b), all the frontier points are also boundary points. The problem is that they have high probability of being chosen for extension in the direction of the obstacles, but most of the times the extension cannot be performed.

In general, boundary points are given more Voronoi bias than they can explore. As a consequence, prohibitively

[1]This name is inspired by actual devices for catching bugs. They can enter easily, but it is hard to escape. The analogy was suggested by James O'Brien in a conversation with James Kuffner.

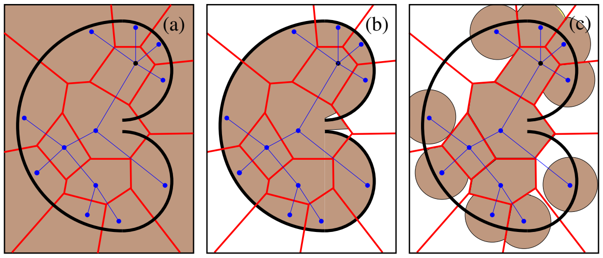


Fig. 3. For a set of points inside a bug trap different sampling domains are shown: (a) regular RRTs sampling domain, (b) visibility region, (c) dynamic domain.

many expensive operations are being performed during the execution of the RRT planners. For example, interpolation between the new configuration and its nearest node in the tree is required at each iteration in the holonomic version of the RRTs. This involves several collision checks being performed along the interpolation path, which may be very expensive in the environments with complicated geometry [10]. Inverse kinematics, together with interpolation, should be performed at each iteration of some RRT-based planner for systems with closed linkages [9]. For nonholonomic planners [6], [25], some integration is also required at each iteration.

Thus, the goal of this paper is to find a way of reducing the number of expensive iterations in RRTs by controlling the Voronoi bias of the nodes in the tree. We define the problem formally in the next subsection.

B. Problem Definition

Let C be an n-dimensional configuration space, and C_{obs} be the set of obstacles in this space. Let V be a set of N collision free points lying inside $C_{free} = C \setminus C_{obs}$ (i.e. the current RRT's nodes), and $\mathcal{D}$ be the Voronoi diagram of V.

Definition 2.1: Let $\mathcal{L}$ be any local method that computes a path $\mathcal{L}(v, v')$ (e.g., the straight line segment) between two given nodes in the tree v and v'. We define the *visibility domain* of a point v for $\mathcal{L}$ as in [31]:

$$Vis_{\mathcal{L}}(v) = \{v' \in C_{free} \text{ such that } \mathcal{L}(v, v') \in C_{free}\}$$

Definition 2.2: For a point $v \in V$ and its Voronoi region $D(v)$ define the *visible Voronoi region* of v to be $O(v) = Vis(v) \cap D(v)$.

Now consider the union of all the visible Voronoi regions $\bigcup_{v \in V} O(v)$, which together comprise the visibility region of the set of points V. A uniform distribution over $\bigcup_{v \in V} O(v)$, which we call a *visibility distribution*, would be more appropriate for the RRT planner than the uniform distribution over the configuration space. Indeed, this distribution does not have bias for exploring toward the obstacles. At the same time it includes the important bias toward the unexplored parts of the configuration space. Examples of the visibility

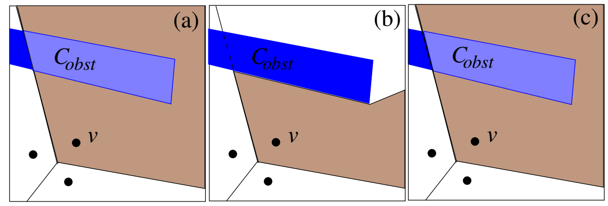


Fig. 4. For a non-boundary point different sampling domains are shown: (a) regular RRTs sampling domain, (b) visible Voronoi region, (c) dynamic domain.

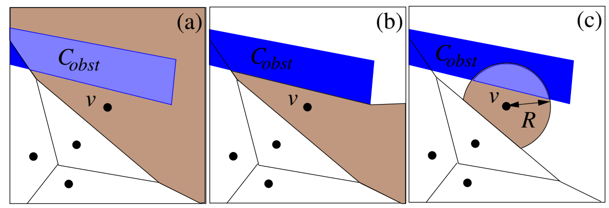


Fig. 5. For a boundary point different sampling domains are shown: (a) regular RRTs sampling domain, (b) visible Voronoi region, (c) dynamic domain.

distribution for a linear local planner and the uniform distribution over all configuration space for the set of points inside the bug trap are shown in Figures 3(a) and 3(b).

Ideally, the distribution over the visible Voronoi diagram should not be uniform. It should concentrate more points inside the narrow corridors and less points in larger open areas of the space. This by itself is a very hard problem, since finding narrow corridors may be harder than solving the original motion planning problem. Producing a uniform distribution from the visible Voronoi regions is already quite a challenging task. In this paper we propose another distribution, which is easier to compute but retains some useful properties of the visibility distribution. The next section describes this approach.

III. Dynamic-Domain RRT Path Planner

Although the visibility distribution provides the necessary bias to the RRT planners, computing it may be expensive. Consider computing the visible Voronoi region $O(v)$ of some node v in the tree. Finding the points in $D(v)$ that are visible from v may require performing interpolation and, therefore, expensive collision detection calls. This is exactly what planners should try to avoid when the interpolation is expensive. Therefore, we consider another distribution which we define as follows.

Definition 3.1: Given a boundary point v at distance at most ϵ from an obstacle in C_{obs}, define the *boundary domain* for v as the intersection of the Voronoi region of v and an n-dimensional sphere of radius R, centered at v.

Definition 3.2: The *dynamic domain of radius* R for the set of points V is the boundary domains of the boundary points combined with the Voronoi regions of all other

```
BUILD_DYNAMIC_DOMAIN_RRT(q_init)
1    T.init(q_init);
2    for k = 1 to K do
3        repeat
4            q_rand ← RANDOM_CONFIG();
5            q_near ← NEAREST_NEIGHBOR(q_rand, T);
6        until dist(q_near, q_rand) < q_near.radius
7        if CONNECT(T, q_rand, q_near, q_new)
8            q_new.radius = ∞;
9            T.add_vertex(q_new);
10           T.add_edge(q_near, q_new);
11       else
12           q_near.radius = R;
13   Return T;
```

Fig. 6. The dynamic domain RRT construction algorithm

points. The uniform distribution over this domain is called the *dynamic domain distribution*.

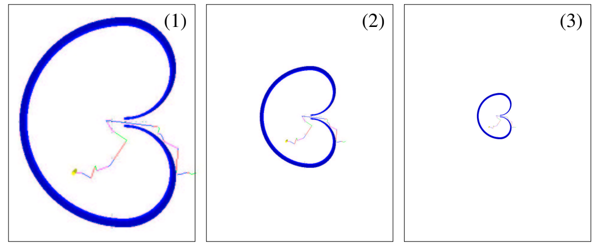
The differences between the original RRT's sampling domain, the visibility region and dynamic domain for a set of points inside a bug trap can be seen on Figure 3. When a point is quite far from the obstacles, its boundary domain is the same as the RRT's sampling domain, that is, the whole Voronoi region (Figure 4). On the other hand, when a point is a boundary point, the boundary domain may be much smaller than both the visible Voronoi region and the whole Voronoi region (Figure 5).

The ϵ parameter should be chosen carefully. It should be sufficiently small, so that the point becomes a boundary point when most of the attempts to interpolate from it fail. On the other hand, it should not be smaller than the resolution of the tree, to reduce the number of redundant nodes. Therefore, ϵ can be naturally chosen as the interpolation step in the connect function of the RRT planner. The radius R is chosen as a multiple of ϵ.

A. Implementation

To obtain the dynamic domain distribution, we generate a distribution from the configuration space C, and then restrict it to the dynamic domain. Given that the original distribution was uniform, the obtained restriction is also uniform. Computing this restriction corresponds to the lines 3-6 in the Figure 6. The radius field of each point stores value R, if it is a boundary point, and value ∞ otherwise. It is important to note, that for this step to be efficient the random configurations should be chosen from the area closely fitting the dynamic domain. In our experiments this area is the smallest bounding box among all boundary domains.

Next, we show how to incorporate sampling from the dynamic domain in the RRT-CONNECT planner. The complete pseudocode is shown on figure 6. The algorithm updates the information about the boundary points on the fly. At the beginning of the exploration of the tree, all points are considered to be non-boundary. As soon as the interpolation from one of the nodes fails (meaning that the distance to the obstacles is at least ϵ, where ϵ is length of the interpolation step) the point becomes a boundary point. This corresponds to the lines 11-12 in the code. The radius field of this point is updated to R. Next time the samples from the Voronoi region of this point are restricted to its boundary domain.



	Dynamic-Domain bi-RRT	bi-RRT
time (1)	0.4 sec	0.1 sec
no. nodes (1)	253	37
CD calls (1)	618	54
time (2)	2.5 sec	379 sec
no. nodes (2)	1607	6924
CD calls (2)	3751	781530
time (3)	1.6 sec	$>$ 80000 sec
no. nodes (3)	1301	–
CD calls (3)	3022	–

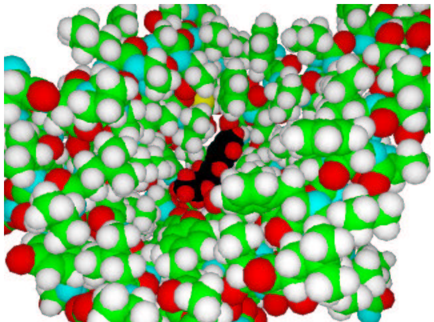
Fig. 7. The goal is to move a point robot out of the two-dimensional bug trap. Results for different environment sizes are shown in (1), (2), (3). Each next environment has 50 times the volume of the previous one.

It is important to note that the dynamic domain RRTs retain the probabilistic completeness of the original RRTs. The argument follows closely to the one presented in [21]. Deterministic, resolution completeness can alternatively be obtained (which results in an RDT) by using a dense sample sequence in the place of the random sequence [24].

Another note is that the proposed method changes its behavior with the size of the radius parameter R. When the radius is chosen to be ∞, this dynamic domain RRT is the same as the original RRT. As the radius value becomes smaller, the behavior of the dynamic domain RRT is more greedy. The tree tends to produce more nodes in the free space, since the bias of the boundary points is reduced. Therefore, efficient nearest neighbor methods adapted to the topology of the configuration space should be used [2]. Unfortunately, we do not have theoretical characterization of the dependence of the performance of the dynamic domain RRT from the radius parameter. However, there is a strong relationship between this parameter and the interpolation step size of the RRT. For all our experiments we set $R = 10\epsilon$.

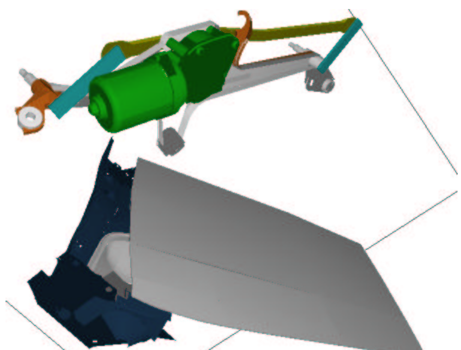
IV. Experimental Results

We have implemented our algorithm in C and incorporated it into the software platform Move3D [32] developed at LAAS. The experiments were performed on a 333 MHz Sunblade 100 running SunOs 5.9 and compiled under gcc 3.3. We have compared the performance



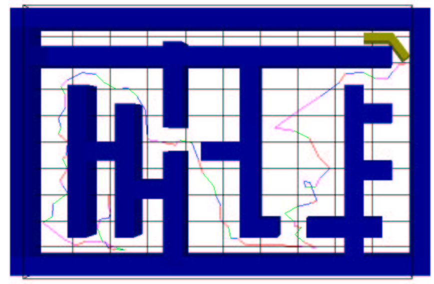
	Dynamic-Domain bi-RRT	bi-RRT
time	70 sec	2926 sec
no. nodes	1358	428
CD calls	47710	1257055

Fig. 8. Molecule example. The task is to compute the pathway of a small molecule to the active site located inside the protein model.



	Dynamic-Domain bi-RRT	bi-RRT
time	217 sec	$>$ 80000 sec
no. nodes	219	–
CD calls	30443	–

Fig. 9. An example from the manufacturing industry. The goal is to find a collision free path dismounting a wiper motor from a car body.



	Dynamic-Domain bi-RRT	bi-RRT
time	161 sec	237 sec
no. nodes	25483	20392
CD calls	604503	464137

Fig. 10. The goal in this example is to move a two link articulated body with 4 degrees of freedom from one corner of the labyrinth to another.

of the bidirectional balanced RRT-CONNECT algorithm and Dynamic-Domain bidirectional RRT-CONNECT. For each of the experiments we show the running times, the number of nodes in the solution trees and the number of the collision detection (CD) calls during the construction process, averaged over 50 runs.

We first show the results obtained for a bug trap in two dimensions (Figure 7). We have picked several different environments to demonstrate the deterioration of the performance of the classical bi-RRT planner with the environment size. In comparison, the average running times of the dynamic domain RRT do not change.

Next experiment was performed on the molecule model shown in Figure 8. Since the molecule is modeled as a 3-d rigid body, the configuration space is 6-dimensional. The task is to compute the pathway of a ligand (i.e. the small molecule displayed in black) to the active site located inside the protein model. The motion planning problem is relatively easy here, since the number of nodes required to find a solution by a regular bidirectional RRT is 400. However, since the collision detection calls are very expensive, the performance of the regular bi-RRT is poor. Therefore, setting up a small radius parameter, which results in a larger number of nodes constructed by the dynamic domain RRT, results in much faster running times.

Next example in Figure 9 is a benchmark from the automotive industry. Automotive industry provides important applications for motion planning, since the industrial companies can verify the manufacturing process and/or the maintainability of the assembly by computing the solution to the motion planning problem. The goal in this example is to find a collision free path dismounting a wiper motor from a car body. This is a real industrial problem which is highly constrained and was solved before with the method described in [10]. The original bi-RRT is not able to solve this problem after running for several days. The dynamic domain RRT solves it on average in several minutes.

The problem shown in Figure 10 is a motion planning problem for a two link articulated body with 4 degrees of freedom. The goal is to move the robot from one corner of the labyrinth to another. The collision checks are relatively cheap in this problem and there are several narrow passages. This problem is easily solved by the original RRT method, however, the dynamic domain RRT is still advantageous.

V. Conclusions and Future Work

We have considered the Voronoi biased exploration strategy of the RRTs and characterized the weaknesses of this strategy when the obstacles in the configuration space are not taken into account and/or the sampling region is inappropriately chosen. We then proposed a general framework for addressing these problems by considering a new sampling strategy based on the visibility region of the nodes in the tree. The new planner adaptively controls the Voronoi bias of the nodes which results in a better exploration. Our experimental results show that the planner successfully solves the problems with constrained geometries as well as other holonomic problems, sometimes by several magnitudes faster than the original RRTs. This suggests that the new planner is suited for a larger set of motion planning problems.

There are several ways to improve the current work. The boundary domains that we consider in this paper are based on spheres. This may sometimes be too constraining, since the boundary domain is smaller than the visible Voronoi region. It also requires tuning the additional parameter of radius of the sphere. One improvement would be to consider other kinds of boundary domains, for example hyperplane-based domains, which are automatically adapted with respect to the distances to the obstacles.

Another important direction is to apply this framework for other constrained motion planning problems such as planning for closed linkages, planning under differential constrains, etc, where both the cost of the iteration computations and the Voronoi bias greatly affect the efficiency of planning algorithms.

ACKNOWLEDGMENTS

We thank the Kineo company (www.kineocam.com) for allowing the publication of the results obtained on the industrial benchmark.

REFERENCES

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, pages 155–168, 1998.

[2] A. Atramentov and S. M. LaValle. Efficient nearest neighbor searching for motion planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 632–637, 2002.

[3] R. Bohlin and L. Kavraki. Path planning using Lazy PRM. In *IEEE Int. Conf. Robot. & Autom.*, 2000.

[4] V. Boor, N. H. Overmars, and A. F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *IEEE Int. Conf. Robot. & Autom.*, pages 1018–1023, 1999.

[5] M. S. Branicky and M. M. Curtiss. Nonlinear and hybrid control via RRTs. In *Proc. Intl. Symp. on Mathematical Theory of Networks and Systems*, 2002.

[6] P. Cheng and S. M. LaValle. Resolution complete rapidly-exploring random trees. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 267–272, 2002.

[7] P. Choudhury and K. Lynch. Trajectory planning for second-order underactuated mechanical systems in presence of obstacles. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, 2002.

[8] J. Cortés and T. Siméon. Sampling-based motion planning under kinematic loop-closure constraints. In *6th International Workshop on Algorithmic Foundations of Robotics*, pages 59–74, 2004.

[9] J. Cortés, T. Siméon, and J.P. Laumond. A random loop generator for planning the motions of closed kinematic chains using PRM methods. In *IEEE Int. Conf. Robot. & Autom.*, 2002.

[10] E. Ferré and J.-P. Laumond. An iterative diffusion method for part disassembly. In *IEEE Int. Conf. Robot. & Autom.*, 2004.

[11] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance and Control*, 25(1):116–129, 2002.

[12] B. Glavina. Solving findpath by combination of goal-directed and randomized search. In *IEEE Int. Conf. Robot. & Autom.*, pages 1718–1723, May 1990.

[13] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed kinematic chains. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, 2000.

[14] C. Holleman and L. E. Kavraki. A framework for using the workspace medial axis in PRM planners. In *IEEE Int. Conf. Robot. & Autom.*, pages 1408–1413, 2000.

[15] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In et al. P. Agarwal, editor, *Robotics: The Algorithmic Perspective*, pages 141–154. A.K. Peters, Wellesley, MA, 1998.

[16] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. Comput. Geom. & Appl.*, 4:495–512, 1999.

[17] S. Kagami, J. Kuffner, K. Nishiwaki, and K. Okada M. Inaba. Humanoid arm motion planning using stereo vision and RRT search. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2003.

[18] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann. Planning collision-free reaching motions for interative object manipulation and grasping. *Eurographics*, 22(3), 2003.

[19] 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. & Autom.*, 12(4):566–580, June 1996.

[20] J. Kim and J. P. Ostrowski. Motion planning of aerial robot using rapidly-exploring random trees with dynamic constraints. In *IEEE Int. Conf. Robot. & Autom.*, 2003.

[21] J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 995–1001, 2000.

[22] F. Lamiraux, E. Ferre, and E. Vallee. Kinodynamic motion planning: connecting exploration trees using trajectory optimization methods. In *IEEE Int. Conf. Robot. & Autom.*, pages 3987–3992, 2004.

[23] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University, Oct. 1998.

[24] S. M. LaValle. *Planning Algorithms*. [Online], 2004. Available at http://msl.cs.uiuc.edu/planning/.

[25] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, Wellesley, MA, 2001.

[26] S. R. Lindemann and S. M. LaValle. Incrementally reducing dispersion by increasing Voronoi bias in RRTs. In *Proc. IEEE International Conference on Robotics and Automation*, 2004.

[27] E. Mazer, J. M. Ahuactzin, and P. Bessière. The Ariadne's clew algorithm. *J. Artificial Intell. Res.*, 9:295–316, November 1998.

[28] C. Pisula, K. Hoff, M. Lin, and D. Manoch. Randomized path planning for a rigid body based on hardware accelerated Voronoi sampling. In *Proc. Workshop on Algorithmic Foundation of Robotics*, 2000.

[29] G. Sánchez and J.-C. Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Int. Symp. Robotics Research*, 2001.

[30] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars. Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *Int. J. Robot. Res.*, 17:840–857, 1998.

[31] T. Siméon, J.-P. Laumond., and C. Nissoux. Visibility based probabilistic roadmaps for motion planning. *Advanced Robotics Journal*, 14(6), 2000.

[32] T. Siméon, J.-P. Laumond., C. van Geem, and J.Cortés. Computer aided motion: Move3d within. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2001.

[33] T. Siméon, J.P. Laumond, J. Cortés, and A.Sahbani. Manipulation planning with probabilistic roadmaps. *Int. J. Robot. Res.*, 23(7-8):739–746, 2004.

[34] P. Svestka and M. H. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *IEEE Int. Conf. Robot. & Autom.*, pages 1631–1636, 1995.

[35] 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 *IEEE Int. Conf. Robot. & Autom.*, pages 1024–1031, 1999.

[36] J. Yakey, S. M. LaValle, and L. E. Kavraki. Randomized path planning for linkages with closed kinematic chains. *IEEE Transactions on Robotics and Automation*, 17(6):951–958, December 2001.