An Improved Random Neighborhood Graph Approach

Libo Yang
Dept. of Computer Science
Iowa State University
Ames, IA 50011 USA
lyang@cs.iastate.edu

Steven M. LaValle
Dept. of Computer Science
University of Illinois
Urbana, IL 61801 USA
lavalle@cs.uiuc.edu

Abstract

As a general framework to determine a collision-free feedback motion strategies, the Random Neighborhood Graph (RNG) approach [19] defines a global navigation function over an approximate representation of the free configuration. In this paper, we improve the RNG approach in several aspects. We present an ANN-accelerated RNG construction algorithm to achieve near logarithmic running time in each iteration of the RNG expansion. Two probabilistic termination conditions of the RNG construction algorithm are presented and analyzed. To help overcome the difficulty of narrow corridors, we also introduce a randomized perturbation algorithm to enhance the sampling quality. Our implementation illustrates a significant performance improvement.

1 Introduction

Determining a collision-free feedback motion strategy is one of the greatest challenges in the design of many robotics systems. Inspired by both the success of randomized path planning techniques [1, 2, 5, 7, 8, 11, 17] and previous work on feedback motion strategies (such as funnels [4, 13], deployments [4, 12], artificial potential fields [9, 10], and navigation functions [16]), we have proposed a randomized framework for generating feedback motion strategies for robots with high degrees of freedom, called the RNG approach, in [19]. The RNG approach can be considered as a method to compute a deployment in complicated configuration spaces. The philosophy of the RNG is similar in some ways to the PRM. We want to capture the topology of the free space in a data structure that can be used for efficient multiple queries. However, the RNG is particularly designed for representing navigation functions.

In previous work [19], we divide the design of a feedback motion strategy into two stages: constructing an approximate representation of $\mathcal{C}_{free}$, and constructing a navigation function over this representation. The first stage is performed only once, while the second stage may be iterated many times for a variety of changing goals. The RNG captures topological information of the free configuration space with overlapping neighborhoods, such as n-dimensional balls or n-dimensional cylinders, on each of which an collision-free partial potential function is defined. A global navigation function can be defined over most of $\mathcal{C}_{free}$ by combining all of the partial potential functions. The RNG construction algorithm terminates if a probabilistic termination condition is satisfied.

In this paper, we improve the RNG approach in several aspects. We present an efficient point location and neighborhood intersection algorithm based on ANN (Approximate Nearest Neighbor Searching) [15] to accelerate the RNG construction. A detailed analysis of the termination conditions of the RNG construction algorithm is presented. A sampling enhancement technique, named randomized perturbation, is introduced and implemented. It increases the sizes of neighborhoods, which reduces the complexity of the RNG.

2 Problem Formulation

Assume that a robot moves in a bounded 2D or 3D world, $\mathcal{W} \subset \mathbb{R}^N$, such that $N = 2$ or $N = 3$. An n-dimensional *configuration vector*, q, captures position, orientation, joint angles, and/or other information for the robot. Let $\mathcal{C}$ be the *configuration space*. Let $\mathcal{A}(q)$ denote the set of points in $\mathcal{W}$ that are occupied by the robot when it is in configuration q. Let $\mathcal{O} \subset \mathcal{W}$ denote a static *obstacle region* in the world. Let $\mathcal{C}_{free}$ denote the set of configurations, q, such that $\mathcal{A}(q) \cap \mathcal{O} = \emptyset$.

The task is to find a motion strategy that uses feedback and guides the robot to a goal configuration from any initial configuration while avoiding collisions. For a given goal, q_{goal}, this can be accomplished by defining a real-valued *navigation function*, $U : \mathcal{C}_{free} \rightarrow \mathbb{R}$ that has a single stable local minimum, which is at q_{goal}. The robot is guided to the goal by following directions given by the negative gradient of U. If q_{goal} is changed, U can be quickly recomputed to guide the robot to the new q_{goal}.

3 The RNG Approach

Assume $\mathcal{C}_{free}$ is bounded. Let $\mu(X)$ denote the measure (or n-dimensional volume) of a subset of $\mathcal{C}_{free}$ (obviously the measure is sensitive to the parameterization of the configuration space). For a given $\alpha \in (0, 1)$, and a probability, P_c, the first phase consists of building a data structure that fills $\mathcal{C}_{free}$ with a set $\mathcal{B} \subset \mathcal{C}_{free}$, such that $\mu(\mathcal{B})/\mu(\mathcal{C}_{free}) \geq \alpha$ with probability P_c. As goals are changed, it must be possible to efficiently recompute a new navigation function. If the system is small-time controllable, a trajectory, not only for basic (holonomic) path planning problems, but also for problems involving differential constraints, can be found by using the navigation function.

Definition 1 *An RNG is an approximate topological representation of $\mathcal{C}_{free}$, defined as:*

- *an undirected graph, $G = (V, E)$, in which V is the set of vertices and E is the set of edges.*
- *a set of neighborhoods $\mathcal{N}$. Let $\mathcal{B}$ be the union of all neighborhoods in $\mathcal{N}$,*
$$\mathcal{B} = \bigcup_{B \in \mathcal{N}} B \ .$$
- *each vertex, $v \in V$, represents an n-dimensional neighborhood, $B_v \in \mathcal{N}$, which lies entirely in $\mathcal{C}_{free}$.*
- *an edge, $e \in E$, exists for each pair of vertices, v_i and v_j, if and only if their neighborhoods intersect, $B_i \cap B_j \neq \emptyset$.*

Obviously, $\mathcal{B}$ is the subset of $\mathcal{C}_{free}$ that is occupied by neighborhoods. $\mathcal{B}$ gives an approximate representation of $\mathcal{C}_{free}$. The following lemma shows that under certain conditions, $\mathcal{B}$ converges to $\mathcal{C}_{free}$ as the number of vertices increases.

Lemma 1 *Assume $\mathcal{C}_{free}$ and B_v are both open sets in an n-dimensional Euclidean space, and let $\mu(X)$ denote the measure of a subset of $\mathcal{C}_{free}$. Let $|V|$ denote the number of vertices in V. If the center of B_v is chosen uniformly and randomly from $\mathcal{C}_{free}$, then*
$$\lim_{|V| \to \infty} \mu(\mathcal{C}_{free} \setminus \mathcal{B}) = 0 \ .$$

Proof: Since unions of open sets are open sets, there must exist a sequence of B_v such that $\mathcal{C}_{free}$ can be represented as the union of all B_v. We want to show that uniformly and randomly choosing the center of B_v from $\mathcal{C}_{free}$ will yield such a sequence. By contradiction, assume $\mathcal{B}$ does not converge to $\mathcal{C}_{free}$ as $|V|$ increases. Then, there must exist a open subset of $\mathcal{C}_{free}$, $T \subset \mathcal{C}_{free}$, that contains no B_v. The probability of B_v falling into T is 0. Thus, the probability distribution of B_v over $\mathcal{C}_{free}$ is not uniform. This contradicts the lemma's condition that the center of B_v is chosen from $\mathcal{C}_{free}$ randomly and uniformly. Therefore, as the number of vertices in G increases, the union of all neighborhoods, $\mathcal{B}$, will converge to C_{free} in measure. ■

4 An ANN Accelerated RNG Algorithm

We grow the RNG iteratively by adding a new node chosen randomly and uniformly in $\mathcal{C}_{free}$. To make sure the RNG will cover $\mathcal{C}_{free}$ efficiently, we only keep a new node if its configuration is outside of $\mathcal{B}$. Figure 1 gives an outline of the algorithm. For a given $\alpha \in (0, 1)$ and $P_c \in (0, 1)$, the algorithm will construct an RNG such that with probability P_c, $\mu(\mathcal{B})/\mu(\mathcal{C}_{free}) \geq \alpha$.

Checking whether $q_{new} \in \mathcal{B}$ is a nontrivial *point location problem.* A naive method yields $O(n)$ running time, assuming the number of nodes in the RNG is n. However, there exist techniques from computational geometry can be exploited to efficiently solve the problem. For example, the random geometric separators technique [14] can provide $O(\lg n)$ expected running time to locate a new configuration and $O(n \lg n)$ expected running time to make edges. In this paper, another approach named ANN (Approximate Nearest Neighbor Searching) [15] is used to accelerate the progress of locating a new configuration and making edges.

The idea of using ANN is motivated by the observation that there are limited number of edges per node in average (about 30) in the RNG. As the RNG grows, the number of edges of a new node is usually reduced because the size of new neighborhood is usually reduced, therefore resulting in less connections. Thus, it is reasonable to narrow the search for point location and neighborhood intersection into a subset of G without losing topological information. Such a subset of G can be defined as w nearest neighborhoods of q_{new} in G. The ANN accelerated RNG construction algorithm is an approximation of the original RNG construction algorithm. Figure 1 shows the ANN accelerated RNG construction algorithm.

```
GENERATE_RNG($\alpha$,$P_c$)
1   G.init($q_{init}$);
2   while (TerminationUnsatisfied(G,$\alpha$,$P_c$) do
3       repeat
4           $q_{new} \leftarrow$ RandomConf(G);
5           $d \leftarrow$ DistanceComputation($q_{new}$);
6           $G' \leftarrow$ NearestNeighbor($q_{new}, w$);
7       until (($d > 0$) and ($q_{new} \notin G'$))
8       $r \leftarrow$ ComputeRadius(d);
9       $v_{new} \leftarrow$G.AddVertex($q_{new}, r$);
10      G.AddEdges($v_{new}, G'$);
11  Return G
```

Figure 1: This algorithm constructs the RNG and determines automatically when to terminate based on estimated coverage of $\mathcal{C}_{free}$. ANN is used to accelerate the RNG construction.

Let G' be a subset of G selected by ANN, which contains w (usually $w < 30$) nearest neighborhoods of q_{new}. Line 6 finds G' for each q_{new}. In line 7, we test if $q_{new} \notin G'$ to approximate if $q_{new} \notin \mathcal{B}$. If true, we expand the RNG. q_{new} is added as a new vertex of G in line 9. Line 10 makes edges between q_{new} and G' to approximate the edges between q_{new} and G. Obviously, there is a little chance that q_{new} is actually inside $\mathcal{B}$ or some edges of q_{new} are missed. However, as we discussed above, the probability of such errors is small and converges to zero as the RNG grows. ANN can find w nearest neighborhoods of q_{new} in near logarithmic time. Therefore, each iteration of adding a new neighborhood into the RNG can be performed in near logarithmic time.

5 Termination Condition

One advantage of the RNG approach is it has a probabilistic termination condition, similar to VisPRM [17]. Some randomized path planning algorithms [1, 2, 7, 8] have to be terminated arbitrarily. Others [3, 11] do not terminate until a path is found. However, the RNG construction algorithm in Figure 1 can decide to terminate by itself based on a statistical estimate of the fraction of $\mathcal{C}_{free}$ that is covered by the RNG.

The volumes of $\mathcal{C}_{free}$ and $\mathcal{B}$, denoted by $\mu(\mathcal{C}_{free})$ and $\mu(B)$, are assumed unknown. Although it is theoretically possible to incrementally compute $\mu(B)$, it is generally too complicated. A probabilistic termination condition can be derived based on the number of samples that fall into $\mathcal{B}$, as opposed to $\mathcal{C}_{free} \setminus \mathcal{B}$. For a given α and P_c, the algorithm will terminate when 100α percent of the volume of $\mathcal{C}_{free}$ has been covered by the RNG with probability P_c.

Let Y be a random variable corresponding to a new configuration in each iteration of the RNG construction algorithm. The experiment of Y has two possible outcomes: either the random configuration, $q_{new} \in \mathcal{B}$, which is a "failure", or $q_{new} \in \mathcal{C}_{free} \setminus \mathcal{B}$, which is a "success". Let $Y = 0$ denote $q_{new} \in \mathcal{B}$, and let $Y = 1$ denote $q_{new} \in \mathcal{C}_{free} \setminus \mathcal{B}$. Since each new configuration is chosen uniformly from $\mathcal{C}_{free}$, then Y has Bernoulli distribution, i.e.,

$$P[Y = 0] = \theta \ \ and \ \ P[Y = 1] = 1 - \theta,$$

in which $\theta = \mu(\mathcal{B})/\mu(\mathcal{C}_{free})$. The following lemma shows that the trials of Y can be used to estimate θ.

Lemma 2 *Let $Y_1, Y_2, \ldots, Y_m$ represent a sequence of trails of Y, and they constitute a random sample. Let $\bar{Y}$ be the sample mean, then $1 - \bar{Y}$ is an unbiased estimator of θ. Moreover, for given fraction α and $T \in (0, 1)$, the probability, P_c, that $1 - \bar{Y} \geq T$ implies $\theta \geq \alpha$ is*

$$P_c = 1 - \alpha^{m+1} \sum_{i=0}^{\lfloor m(1-T) \rfloor} \binom{m}{i} \left(\frac{1}{\alpha^i(m-i+1)} - \frac{1}{m+1} \right). \tag{1}$$

We call P_c the **confidence level**.

Proof: The first part of the lemma that $1 - \bar{Y}$ is an unbiased estimator of θ is obvious from the fact that Y has the Bernoulli distribution. Now we prove the second part of the lemma. First, consider the error of our hypothesis. There are two kinds of errors: rejection of the null hypothesis if it is true is called a **type I** error; acceptance of the null hypothesis if it is false is called a **type II** error. More precisely, $1 - \bar{Y} < T$ if $\theta \geq \alpha$ causes a type I error. $1 - \bar{Y} \leq T$ if $\theta < \alpha$ causes a type II error. Let P_I and P_{II} represent the probability that a type I error and a type II error occurs, respectively. Then, we have $P_I = P\left[1 - \bar{Y} < T \ ; \ \theta \geq \alpha\right]$ and $P_{II} = P\left[1 - \bar{Y} \geq T \ ; \ \theta < \alpha\right]$. Note that, $P_c = 1 - P_{II}$.

Let $y_1, y_2, \ldots, y_m$ represent the outcomes of a sequence of experiments of Y. Let X be a random variable that denotes the number of trials of Y at which the k successes occur. X has a binomial distribution. Then, P_{II} can be written as:

$$\begin{aligned} P_{II} &= P\left[(1 - \bar{Y}) \geq T \ ; \ \theta < \alpha\right] \\ &= \alpha^{m+1} \sum_{i=0}^{\lfloor m(1-T) \rfloor} \binom{m}{i} \left(\frac{1}{\alpha^i(m-i+1)} - \frac{1}{m+1} \right). \end{aligned}$$

Therefore, we have,

$$P_c = 1 - \alpha^{m+1} \sum_{i=0}^{\lfloor m(1-T) \rfloor} \binom{m}{i} \left(\frac{1}{\alpha^i(m-i+1)} - \frac{1}{m+1} \right).$$

Thus, we are done. ■

Equation (1) implies that P_c increases as m increases. For given fraction α and P_c, Lemma 2 provides a method to estimate whether the fraction of $\mathcal{B}$ over $\mathcal{C}_{free}$ is at least α with confidence level at least P_c. However, it is not the only method to do so. The following lemma presents another way to estimate θ.

Lemma 3 *Let $Y_1, Y_2, \ldots, Y_m$ represent a sequence of trails of Y, and they constitute a random sample. Assume the first success occurs at Y_m. Then, $1 - \bar{Y}$ is an unbiased estimator of θ. Moreover, for given fraction α and $M > 0$, the probability, P_c, that $m \geq M$ implies $\theta \geq \alpha$ is*

$$P_c = 1 - \frac{\alpha^M}{M}. \tag{2}$$

Both Lemma 2 and Lemma 3 provide a method to estimate θ, based on the different statistical hypothesises. Two termination conditions can be derived from them respectively:

- h_1: terminate based on m trials of Y, at which k success have been counted.
- h_2: terminate based on m successive failures followed by the first success.

The following proposition shows how h_1 and h_2 help to make the termination decision.

Proposition 1 *For given fraction α and P_c, the probability of the fraction $\theta \geq \alpha$ is at least P_c, i.e.,*

$$P[\theta \geq \alpha] \geq P_c,$$

if the termination condition satisfies either one of the following:

- *h_1:*
$$P_c \geq 1 - \alpha^{m+1} \sum_{i=0}^{k} \binom{m}{i} \left(\frac{1}{\alpha^i(m-i+1)} - \frac{1}{m+1} \right) \tag{3}$$
- *h_2:*
$$m \geq \frac{\ln(1 - P_c)}{\ln \alpha} - 1 \tag{4}$$

Although both h_1 and h_2 are feasible termination conditions in practice, they have different performance. The following proposition shows for the same $\bar{Y}$, h_1 has better or equal confidence level than h_2.

Proposition 2 *For the same $\bar{Y}$, the termination condition h_1 has higher confidence level than h_2.*

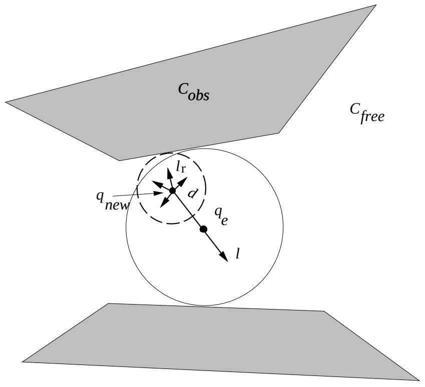

Figure 2: Enhance q_{new} by randomized perturbations.

According to Proposition 2, it seems h_1 is always better than h_2. However, to achieve the same $\bar{Y}$, h_1 needs more samples than h_2. This yields more computation time. A better strategy is to terminate the algorithm if either h_1 is satisfied or h_2 is satisfied. During execution, we open a sample window with a fixed window size m in the sample process. Inside the window, k successes are observed. In each iteration of adding a new configuration, we advance the sample window to include the new experiment and apply the termination condition h_1 on it. This process continues until a success is found. We count the number of successive failures before this success, and apply the termination condition h_2 on it. The algorithm terminates if either one of h_1 and h_2 is satisfied. In general, if a larger m is chosen, then a more accurate decision has been made, but more computation time is needed. Notice, inside the sample window, θ is not fixed in practice. Once a new configuration is found, a new neighborhood is added into $\mathcal{B}$, which increases θ. However, if the α and P_c are close to 1, such an increase is negligible and is ignored.

6 Sampling Enhancement

Sometimes, choosing a new configuration uniformly and randomly from $\mathcal{C}_{free}$ is not enough, especially when dealing with narrow corridors. In Figure 2, if q_{new} is close to the obstacles, the distance d is small, resulting a small neighborhood and less efficiency. The volume of a new neighborhood is determined based on distance computation information. A larger d implies a larger neighborhood. The goal is to find a sampling scheme to maximize d. One such sampling scheme is sampling onto the medial axis [6, 18]. However, it involves finding the closest point of the sampled node in $\mathcal{C}_{obs}$, which is expensive.

In this paper, we present a new approach based on randomized perturbation. The randomized perturbation algorithm has two steps: selecting the right direction and optimizing in this direction. The idea is similar to retraction [18]; from a given q_{new}, randomly choose several directions and pick up the one that has most significant gradient in terms of d, then optimize q_{new} in this direction. The randomized perturbation algorithm can be treated as an approximation of sampling onto the medial axis.

```
RANDOM_PERTURB($q_{new}$)
1   $q_e \leftarrow q_{new}$, $d \leftarrow$ DistanceComputation($q_{new}$);
2   **if** $d < d_{tiny}$ **then**
3       Grad $\leftarrow$ 0;
4       **for** $i \leftarrow 0$ to $k$ **do**
5           $\vec{l_i} \leftarrow$ RandomDirection($q_{new}$);
6           **if** Gradient($\vec{l_i}$) $>$ Grad **then**
7               Grad $\leftarrow$ Gradient($\vec{l_i}$), $\vec{l} \leftarrow \vec{l_i}$;
8       Terminate $\leftarrow$ **True**;
9       **while** (Terminate) **do**
10          $q_t \leftarrow$ Enhancement($\vec{l}, q_{new}, q_e$);
11          $d_t \leftarrow$ DistanceComputation($q_t$);
12          **if** $d_t > d$ and $\|q_{new} - q_t\| < d_t$ **then**
13              $q_e \leftarrow q_t$, $d \leftarrow d_t$, Terminate $\leftarrow$ **False**;
14  Return $q_e$.
```

Figure 3: The randomized perturbation algorithm.

Figure 3 shows the randomized perturbation algorithm.

Lines 3-7 find the most significant direction, $\vec{l}$, along which d has the most significant increase, among k randomly generated directions. Usually, using a larger k will result a better $\vec{l}$. As k increases, $\vec{l}$ converges to the optimal value. However, a larger k also means more computation time for collision checking. In practice, it is sufficient to choose k between 5 and 10.

Lines 8-13 optimize q_{new} following $\vec{l}$. The enhancement procedure terminates and returns an enhanced configuration, q_e, until d no longer increases or the neighborhood centered at q_e no longer contains q_{new}. The latter guarantees the enhanced neighborhood covers the most of space that is covered by the original neighborhood. Line 10 finds a better configuration, q_t, during each iteration. q_t is choosing in the following way: we start with a q_t that is far away from q_{new} in the direction $\vec{l}$; if it does not satisfy the termination conditions, the next candidate is the one at the middle of q_{new} and q_t. Such an optimization procedure is very efficient in practice because only logarithmic time is needed to obtain the q_e.

To avoid too many edges that may potentially be generated, we limit the randomized perturbations algorithm to the sample that has small d. Line 2 corresponds to such a preselecting procedure. We use d_{tiny} as a threshold to represent the smallest d with which a simple does not need to be enhanced. d_{tiny} is chosen according to different applications. A smaller d_{tiny} results a better sampling strategy but more edges result in G.

The randomized perturbation algorithm can be treated as an approximation of sampling onto the medial axis. Sampling onto the medial axis guarantees that the enhanced neighborhood includes the entire original neighborhood. It also guarantees that the enhancement follows the most efficient direction. The randomized perturbation algorithm follows the direction that may not be the most efficient one. It guarantees that most of the original neighborhood has been contained by the enhanced

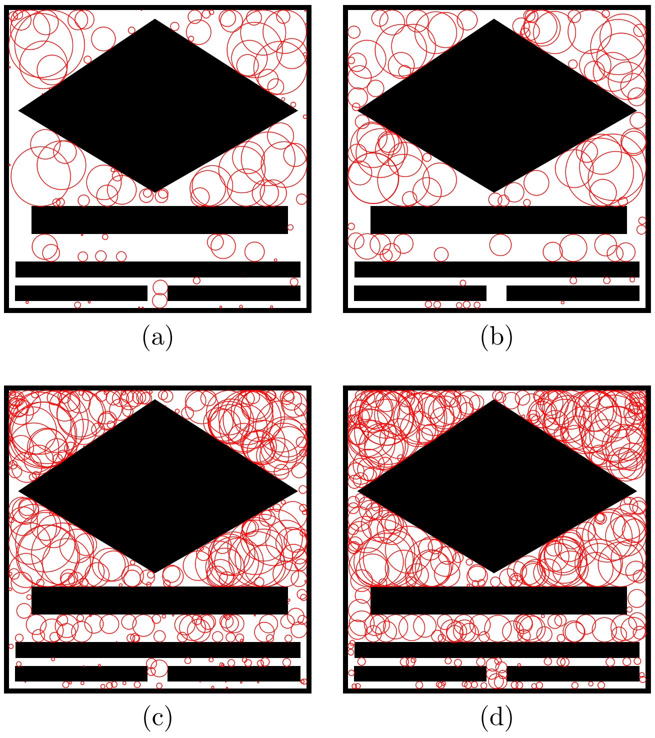

Figure 4: The comparison of randomized perturbation algorithm with no sampling enhancement in a 2D point robot example: (a), (c) show sampling without enhancement with 100 and 300 nodes respectively; (b), (d) show sampling with randomized perturbations with 100 and 300 nodes respectively.

neighborhood. However, the randomized perturbation algorithm is much time efficient than sampling onto the medial axis.

Figure 4 illustrates the advantage of the randomized perturbation algorithm through a simple 2D point robot example. Most of the tiny balls have been enlarged by using randomized perturbation. The coverage of $\mathcal{C}_{free}$ has also been increased significantly. After adding 300 nodes to G, the maximum number of trials to put a new sample is 16 if using randomized perturbation, but only 7 if without enhancement. Since the random perturbation is an approximation of sampling onto the medial axis, and has no guarantee that every tiny ball will be enlarged, there are still a few tiny balls remaining in Figures 4.b and 4.d.

7 An Implementation with Examples

We have implemented the RNG construction algorithm in Gnu C++ using the LEDA library on a Pentium III 500 Mhz PC running Linux. A variety of experiments have been performed for robots in 2D and 3D environments and up to six degrees of freedom. Figure 5 shows robot trajectories of some of our examples. The computation times of constructing those RNGs are listed in Figure 6.

The RNG construction time can be improved in many different ways. We can observe significant improvement

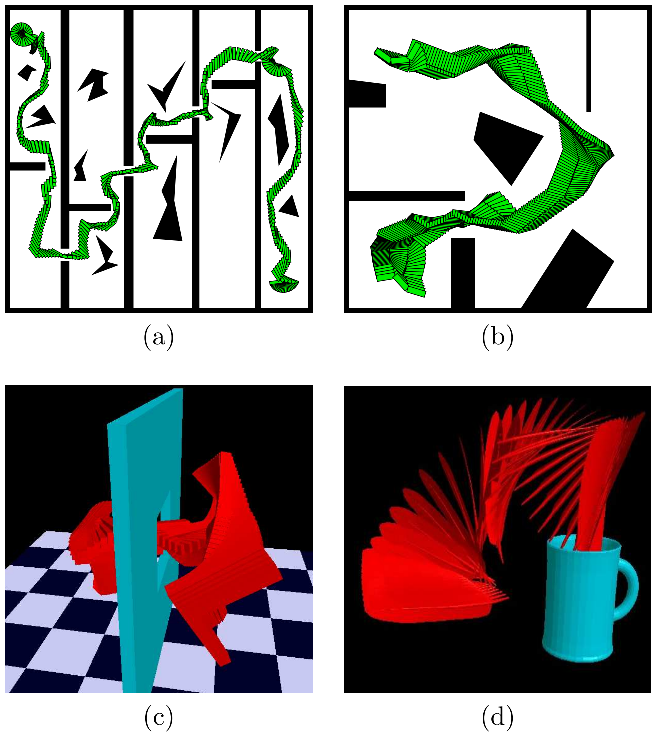

Figure 5: (a) A 3D RNG for a 2D rigid robot; (b) a 5D RNG for an 2D articulated robot; (c) a 6D RNG for a 3D L-shape robot; (d) a 6D RNG for a 3D rigid robot.

	α/P_c	# of Fails	# of Nodes	# of Edges	Construction Time(s)
5.a	.95/.99	100	20000	78676	9747.6/87.0
5.d	.90/.65	10	12000	74432	4614.3/25.0
5.c	.90/.88	20	83922	1792958	66444.7/168.8
5.d	.90/.99	40	6059	335630	13318.0/13.2

Figure 6: The improvement of applying ANN for different examples. Construction time shows without applying ANN versus ANN.

in the RNG construction time by applying the ANN algorithm. The computation time can also be improved by weakening the requested percentage of coverage. Notice that the termination conditions we used imply different combinations of the percentage of coverage and the confidence level associated with it. Suppose h_1 is used. For the given α and P_c, we choose the sample window size, m, and the expected number of successes inside the sample window, k, based on (3). Once the algorithm is terminated according to m and k, it implies that, not only the percentage of coverage α is achieved with the confidence level P_c, but also a percentage of coverage that is higher than α, is achieved with a confidence level that is less than P_c. A similar result can be obtained for h_2.

The sampling enhancement techniques can improve the performance of the RNG in narrow corridors. By applying the randomized perturbation algorithm from Section

6, we expect fewer nodes in the RNG and less computation times. Figure 7 shows the comparison of applying randomized perturbation algorithm and sampling without enhancement for several examples. Notice that the purpose of sampling enhancement is to improve the sampling quality in the narrow corridors, not for the samples in the open spaces. Figure 5.c involves a narrow corridor, and Figure 5.d is relatively easy. This is why a significant performance improvement is observed in Figure 5.c, while there is not much change in Figure 5.d.

	α/P_c	# of Nodes (a)	# of Nodes (b)	Improvement factor
5.c	.90/.88	83922	11073	7.56
5.d	.90/.99	6059	4952	1.22

Figure 7: Randomized perturbations yield significant improvement. (a) corresponds to no enhancement. (b) corresponds to applying randomized perturbations.

8 Discussion

The RNG approach [19] establishes a general framework for planning feedback motion strategies, which is expected to have applications to robotics systems that involve changing environments, moving obstacles, unexpected obstacles, sensing uncertainties, and uncertainties in control. In this paper, we have improved the RNG approach in several aspects. Although there are other existing algorithms for efficient point location, and neighborhood intersections, an ANN accelerated RNG construction algorithm achieves near logarithmic running time in each iteration of the RNG expansion. We have demonstrated two different types of termination conditions. By taking the advantage of both of them, a hybrid termination condition has been implemented, which terminates the RNG construction algorithm more accurately and efficiently. The sampling enhancement algorithm, a random perturbation algorithm, helps improve the performance of the RNG by yielding larger neighborhoods.

Our implementation and experimental results have been encouraging. However, although the random perturbation algorithm helps to improve the sampling quality, our method is expected to suffer, along with other randomized planning methods, from the narrow passage problem [7]. Randomized path planning techniques, such as probabilistic roadmaps [8] and rapidly-exploring random trees [11] might also be helpful to locate locations for balls that will improve RNG performance in narrow passages.

Acknowledgments

This work was funded in part by NSF CAREER Award IRI-9875304 (LaValle). We thank Ahmad Massoud and Sunder Sethuraman for their helpful comments and discussions.

References

[1] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 113–120, 1996.

[2] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robot. Res.*, 10(6):628–649, December 1991.

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

[4] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behavious. *Int. J. Robot. Res.*, 18(6):534–555, 1999.

[5] D. Challou, D. Boley, M. Gini, and V. Kumar. A parallel formulation of informed randomized search for robot motion planning problems. In *IEEE Int. Conf. Robot. & Autom.*, pages 709–714, 1995.

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

[7] 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.

[8] 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.

[9] O. Khatib. *Commande dynamique dans l'espace opérational des robots manipulateurs en présence d'obstacles.* PhD thesis, Ecole Nationale de la Statistique et de l'Administration Economique, France, 1980.

[10] D. E. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *Proc. IEEE Int. Conf. Robot. & Autom.*, pages 1–6, 1987.

[11] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1999.

[12] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic systhesis of fine-motion strategies for robots. *Int. J. Robot. Res.*, 3(1):3–24, 1984.

[13] M. T. Mason. The mechanics of manipulation. In *Proc. IEEE Int. Conf. Robot. & Autom.*, pages 544–548, 1985.

[14] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *Journal of the ACM*, 44(1):1–29, January 1997.

[15] D. Mount and S. Arya. ANN: Library for Approximate Nearest Neighbor Searching. Available at http://www.cs.umd.edu/ mount/ANN/, 1998.

[16] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Trans. Robot. & Autom.*, 8(5):501–518, October 1992.

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

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

[19] L. Yang and S. M. LaValle. A framework for planning feedback motion strategies based on a random neighborhood graph. In *IEEE Int. Conf. Robot. & Autom.*, 2000.