Narrow Passage Sampling for Probabilistic Roadmap Planning

Zheng Sun, David Hsu, Tingting Jiang, Hanna Kurniawati, and John H. Reif

Abstract— **Probabilistic roadmap (PRM) planners have been successful in path planning of robots with many degrees of freedom, but sampling narrow passages in a robot's configuration space remains a challenge for PRM planners. This paper presents a hybrid sampling strategy in the PRM framework for finding paths through narrow passages. A key ingredient of the new strategy is the *bridge test*, which reduces sample density in many unimportant parts of a configuration space, resulting in increased sample density in narrow passages. The bridge test can be implemented efficiently in high-dimensional configuration spaces using only simple tests of local geometry. The strengths of the bridge test and uniform sampling complement each other naturally. The two sampling strategies are combined to construct the hybrid sampling strategy for our planner. We implemented the planner and tested it on rigid and articulated robots in 2-D and 3-D environments. Experiments show that the hybrid sampling strategy enables relatively small roadmaps to reliably capture the connectivity of configuration spaces with difficult narrow passages.**

Index Terms— **Robotics, motion planning, randomized algorithm, random sampling, probabilistic roadmap planner.**

I. Introduction

During the past decade, probabilistic roadmap (PRM) planning [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] has emerged as a powerful framework for path planning of robots with many degrees of freedom (dofs). The main idea of a classic PRM planner [7] is to sample at random a robot's configuration space and connect the sampled points to construct a *roadmap* graph that captures the connectivity of the free space, the collision-free subset of the configuration space. Due to its efficiency and simplicity, PRM planners have found many applications in addition to robotics, including virtual prototyping and computational biology (see, *e.g.*, [11], [12], [13], [14], [15]).

Despite the success of PRM planners, path planning for many-dof robots is difficult. Several instances of the problem have been proven to be PSPACE-hard [16] or even undecidable [17], [18]. It is unlikely that sampling, the key idea behind PRM planners, can overcome such difficulty entirely. Indeed, narrow passages in the configuration space pose significant difficulty for PRM planners. Intuitively, narrow passages are small regions whose removal changes the connectivity of the configuration space. We can also give formal characterizations [19], [6] using the notion of *visibility sets*, where two points in the configuration space are considered visible to each other if they can be connected by a collision-free straight-line path. To capture the connectivity of the configuration space well, PRM planners must sample in the narrow passages. This is difficult, because narrow passages have small volumes, and the probability of sampling from small sets is low.

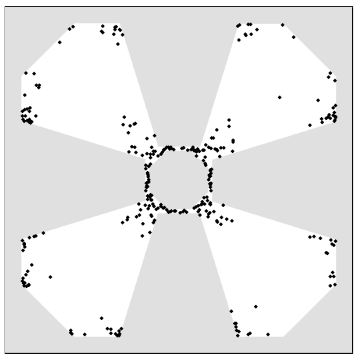


Fig. 1. An example of sample points generated with the bridge test. In this and all later figures, shaded regions indicate obstacles. Black dots indicate sample points.

In this paper, we propose a hybrid sampling strategy in the PRM framework in order to find paths through narrow passages efficiently. Our goal is to build a good roadmap by sampling a small number of well-placed points from the configuration space. We pay a slightly higher computational cost in sampling than simpler alternatives, but our roadmap requires much fewer points to capture the connectivity of the configuration space, thus saving a lot of time in checking collision-free connections between the sampled points.

A key ingredient of our new strategy is the *bridge test*, a specialized sampling strategy for narrow passages. It rejects a large fraction of samples in unimportant parts of a configuration space, thus resulting in increased sample density in narrow passages. In a bridge test, we check for collision at three sampled points: the two endpoints and the midpoint of a short line segment s. If the two endpoints are in collision and the midpoint is collision-free, the midpoint is accepted as a new node in the roadmap graph being constructed. We

This work was supported in part by NSF ITR grants EIA-0086015 0326157 and CCR-0086013, in part by NSF QuBIC grants EIA-0218376 and EIA-0218359, in part by DARPA/AFSOR contract F30602-01-2-0561, in part by RGC grant HKBU2107/04E, and in part by NUS grant R252-000-154-112/101.

Z. Sun is with the Department of Computer Science, Hong Kong Baptist University, Kowloon, Hong Kong (e-mail: sunz@comp.hkbu.edu.hk).

D. Hsu and H. Kurniawati are with the Department of Computer Science, National University of Singapore, 3 Science Drive 2, Singapore 117543, Singapore (e-mail: {dyhsu, hannakur}@comp.nus.edu.sg).

T. Jiang and J.H. Reif are with the Department of Computer Science, Duke University, Box 90129, Durham, NC 27708, USA (e-mail: {ruxu,reif}@cs.duke.edu).

Preliminary results of this work have been presented in the Int. Conf. on Robotics & Automation, 2003.

call this a bridge test, because the line segment s resembles a bridge: the endpoints of s, located inside obstacles, act as piers, and the midpoint hovers over the free space.

The bridge test saves computation time by filtering out those sampled points that are unlikely to contribute to an improved roadmap. For a point inside a narrow passage, building *short* bridges through it is easy, due to the geometry of narrow passages; for a point in the middle of wide-open free space, doing so is much more difficult. By favoring short bridges, we filter out many sampled points in wide-open free space, as most of these points do not improve the connectivity of the roadmap. At the same time, points inside narrow passages easily pass the bridge test and are retained in the roadmap to improve its connectivity. See Fig. 1.

The bridge test uses only collision checking as a primitive operation and does not require complex geometric processing in the configuration space. It is simple to implement and can be easily applied to high-dimensional configuration spaces.

Since the bridge test focuses almost solely on the narrow passages, it may fail to sample an adequate number of points to cover the *entire* free space [19]. Interestingly, the difficulty encountered by the bridge test can be overcome by uniform sampling, which tends to place many samples in wide-open free space. The strengths of the bridge test and the uniform sampler complement each other naturally, and the two sampling strategies are combined to produce a hybrid sampling strategy to achieve better results.

In the following, Section II reviews related work. Section III gives an overview of our planner. Sections IV and V present the bridge test and show how to combine it with uniform sampling to construct a hybrid sampling strategy. Section VI discusses practical implementation issues. Section VII reports experiments with our planner on rigid and articulated robots in 2-D and 3-D environments. Section VIII discusses the limitation of the bridge test and possible generalization. Section IX summarizes the main results.

II. Related Work

The difficulty posed by narrow passages and its importance were noted in early work on PRM planners (see, *e.g.*, [7]) and were later articulated in [20]. Several sophisticated sampling strategies can alleviate this difficulty, but a satisfactory answer remains elusive.

One possibility is to sample more densely near obstacle boundaries [1], [3], [21] because points in narrow passages lie close to obstacles. The Gaussian sampler [3] is a simple, efficient algorithm that uses this idea. However, in some cases, many points near obstacle boundaries lie far away from narrow passages and do not help in improving the connectivity of roadmaps. So despite the improvement, sampling near obstacle boundaries may waste many samples in uninteresting regions (see Fig. 3). In some special cases, the Gaussian sampler can be extended to reduce the number of wasted samples by paying a higher computational cost.

Other approaches to narrow passage sampling include dilating the free space [20] and retracting to the medial axis of free space [22]. Both require geometric operations that are expensive to implement in high-dimensional configuration spaces. To reduce the computational cost, various approximation techniques have been proposed [5], [23], [24]. The visibility-based PRM [10] is related to the narrow passage problem. It tries to reduce the number of unnecessary milestones by checking their visibility.

Some of the above approaches and others are compared through systematic experiments [25], [26]. The comparison indicates that the various approaches all have their own strengths for different situations. Thus it is natural to combine them [27], [28], [29]. This idea, which we call hybrid sampling, is also used in the work here.

III. Overview of the Planner

The configuration of a robot with n dofs can be represented as a point in an n-dimensional space $\mathcal{C}$, called the *configuration space*. A configuration q is *free*, if the robot placed at q does not collide with the obstacles or with itself. We define the free space $\mathcal{F}$ to be the set of all free configurations in $\mathcal{C}$, and define the obstacle space $\mathcal{B}$ to be the complement of $\mathcal{F}$: $\mathcal{B} = \mathcal{C} \backslash \mathcal{F}$.

A classic multi-query PRM planner proceeds in two stages. In the first stage, it randomly samples in $\mathcal{F}$ a set of points, called *milestones*. It uses the milestones as nodes to construct a graph G, called a *roadmap*, by adding an edge between every pair of milestones that can be connected via a simple collision-free path, typically, a straight-line segment. After the roadmap has been constructed, multiple queries can be answered quickly in the second stage. Each query consists of an initial configuration s and a goal configuration t, and asks for a collision-free path connecting s and t. The planner first finds two milestones s' and t' in the roadmap G such that s (t, respectively) and s' (t', respectively) can be connected by a collision-free path in $\mathcal{C}$, and then searches for a path in G between s' and t'.

In this paper, we follow this general framework, but address mainly the first stage, roadmap construction. Methods for the second stage are well-understood [7], [25].

An important property of a good roadmap G is *coverage*: for any given (initial or goal) configuration $q \in \mathcal{F}$, there is a collision-free path between q and a milestone in G with high probability. This implies that the milestones in G collectively "covers" a significant portion of $\mathcal{F}$. Another important property is *connectivity*. The roadmap G should capture the connectivity of the underlying free space $\mathcal{F}$ that it represents: any two milestones in the same connected component of $\mathcal{F}$ should also be connected by a path in G. Otherwise the planner would give many false negative answers.

A main difficulty in constructing good roadmaps results from narrow passages in $\mathcal{F}$. Narrow passages are small regions whose removal changes the connectivity of $\mathcal{F}$: they may change the way different regions are connected, or even change the number of connected components. To capture the connectivity of $\mathcal{F}$ in the roadmap, it is essential to sample milestones in narrow passages. This, however, is difficult because of their small volumes. Any volume-based sampling distribution is likely to fail. In particular, the uniform distribution does not work well. Furthermore, when dealing with many-dof

robots, we do not have an explicit representation of $\mathcal{F}$ and cannot locate narrow passages directly by processing the global geometry of $\mathcal{F}$.

Our goal is to build a good roadmap by sampling a small number of well-placed milestones. To obtain milestones in narrow passages, we pay a higher cost for sampling a milestone than simpler methods such as uniform sampling. However, the intuition is that our roadmap would require much fewer milestones to cover $\mathcal{F}$ and capture its connectivity, thus saving a lot of time in checking collision-free connections between the milestones.

More precisely, the running time T of roadmap construction is given by

$$T = N_{\text{mil}} \cdot T_{\text{mil}} + N_{\text{con}} \cdot T_{\text{con}},$$

where

T_{mil}: the average cost of sampling a milestone,
N_{mil}: the number of milestones in the roadmap,
T_{con}: the average cost of checking collision-free connections between two milestones,
N_{con}: the number of calls to check collision-free connections between two milestones.

To illustrate the potential benefits of our approach, let us look at a numerical example. Assume that $T_{\text{mil}} = t$ and $T_{\text{con}} = 10t$, as the cost of checking collision-free connections between two milestones is much higher than that of sampling a milestone. Assume also that $N_{\text{mil}} = n$, and every milestone is checked for connection with two nearby milestones, *i.e.*, $N_{\text{con}} = 2 \cdot N_{\text{mil}} = 2n$. The total running time is then $T = n \cdot t + 2n \cdot 10t = 21nt$. Now suppose that we pay a higher cost in sampling, but are able to reduce the number of milestones needed. For example, $T_{\text{mil}} = 100t$, and $N_{\text{mil}} = n/10$. Then we have the total running time $T = n/10 \cdot 100t + 2n/10 \cdot 10t = 12nt$, which is roughly half of the original running time. By paying a higher cost T_{mil}, we reduce the number of milestones needed in the roadmap and thus reduce N_{con}. Since T_{con} is usually much larger than T_{mil}, the cost of checking connections between milestones dominates, and reducing N_{con} results in reducing the total running time. Although the numerical values in this example are chosen for the purpose of illustration and should not be taken literally, the intuition behind holds more generally and is supported by the experiments (see Section VII). However, we must be careful in balancing the cost and the benefit. If the increase in T_{mil} is much higher than the decrease in N_{con}, the benefit of our approach will be diminished. Therefore T_{mil} should be kept small, if possible.

The sampling distribution that we use for our planner is a weighted mixture of π_{B}, the distribution generated by the bridge test, and π_{U}, the uniform distribution. We describe how to construct π_{B} and combine the two distributions in the next two sections.

After sampling a new milestone q, our planner tries to connect q to nearby milestones via collision-free straight-line paths. Like other PRM variants, our planner tries to connect two milestones only if they lie in different connected components of the roadmap G and the distance between them is smaller than a threshold, according to a suitable distance metric. However, when the size of G is large, there may still be too many milestones whose distances to q are less than the threshold. So the planner tries to connect q with only the K nearest such milestones, where K is a fixed constant. The intuition is that if the planner cannot establish connections from q to the K nearest milestones, most likely it will not be able to establish connections to other milestones further away.

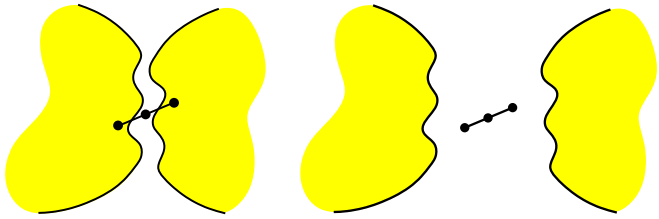


Fig. 2. Building short bridges is much easier in narrow passages (left) than in wide-open free space (right).

IV. The Bridge Test

A. The Algorithm

The bridge test is designed to boost the sample density in narrow passages using only simple tests of local geometry. It is based on the following observation. A narrow passage in an n-dimensional configuration space has at least one restricted direction v such that small perturbations of the robot's configuration along v result in collision of the robot with obstacles. Therefore, for a collision-free configuration q in a narrow passage, it is easy to sample at random a short line segment s through q such that the endpoints of s lie in the obstacles in $\mathcal{C}$ (Fig. 2*a*). The line segment s is called a *bridge*, because it resembles a bridge across the narrow passage. We say that a point $q \in \mathcal{F}$ passes the bridge test, if we succeed in obtaining such a segment s through q. For convenience, we also say that s passes the bridge test. Clearly building *short* bridges is much easier in narrow passages than in wide-open free space. By favoring shorter bridges over longer ones, we increase the chance of accepting points in narrow passages (Fig. 2).

To sample a new milestone using the bridge test, we pick a line segment s from $\mathcal{C}$ by choosing its endpoints at random and test whether s passes the bridge test. If so, we add the midpoint of s to the roadmap G as a new milestone. We call this algorithm Randomized Bridge Builder (RBB). The details of RBB are shown below.

Algorithm 1 Randomized Bridge Builder (RBB).

1. **repeat**
2. Pick a point x from $\mathcal{C}$ uniformly at random.
3. **if** CLEARANCE(x) returns FALSE **then**
4. Pick a point x' in the neighborhood of x according to a suitable probability density λ_x.
5. **if** CLEARANCE(x') returns FALSE **then**
6. Set q to be the midpoint of line segment $\overline{xx'}$.
7. **if** CLEARANCE(q) returns TRUE **then**
8. Insert q into G as a new milestone.

In lines 3, 5, and 7, RBB calls the function CLEARANCE to test whether a point in $\mathcal{C}$ is collision-free.

To perform the bridge test, RBB uses only a single geometric primitive, CLEARANCE, which can be implemented efficiently using a collision detection algorithm (see, *e.g.*, [30], [31]). The bridge test is purely local and does not require processing the global geometry of $\mathcal{C}$.

RBB pays a higher cost to sample a milestone than simpler alternatives such as uniform sampling, because of two reasons. First, it takes three calls to CLEARANCE for RBB to accept a milestone, one for the milestone itself and two for the endpoints of the bridge passing through the milestone. Second, it rejects a large fraction of free configurations that are normally accepted by other sampling strategies. However, RBB increases the sample density in narrow passages, which are critical in capturing the connectivity of the free space. For configuration spaces with difficult narrow passages, it usually leads to smaller roadmaps and saves lots of computation time in checking collision-free connections between milestones, an operation that is usually more expensive than the additional cost that RBB spends in sampling milestones.

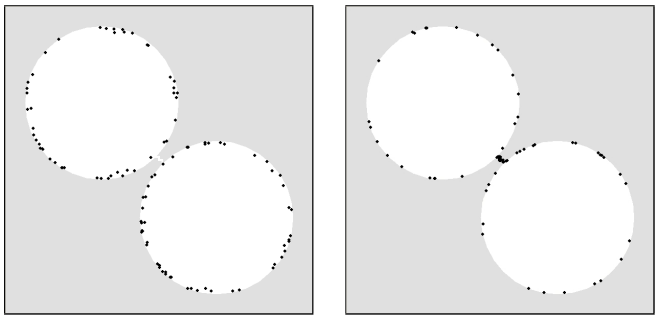
B. Choosing the Probability Density λ_x

The density function λ_x (Algorithm 1, line 4) determines how frequently a bridge of particular length is chosen for a test at the point x. Short bridges are preferred over longer ones in order to increase the probability of sampling in narrow passages. We choose λ_x to be a radially symmetric Gaussian with its center at x and a small standard deviation σ. To be specific, let $N(\sigma)$ denote the univariate Gaussian distribution with mean 0 and standard deviation σ. We sample a value randomly and independently for each dimension of $\mathcal{C}$ according to $N(\sigma)$, shifted to center at the corresponding coordinate of x. The density function λ_x is then the product of these independent univariate Gaussians. Other ways to construct radially symmetric Gaussian are also possible. Finally the parameter σ depends on the width of narrow passages that we want to capture. The best value for σ is problem-specific, and we discuss this issue further in Section VI-B.

C. Analysis of the Sampling Distribution

To calculate probability density π_{B} of the milestones created by Algorithm 1, let us first define X and X' to be two random variables, representing respectively the two endpoints of a bridge. The first endpoint X is distributed uniformly over the set of configuration-space obstacles $\mathcal{B}$. So the density $f_X(x)$ is non-zero if and only if x lies in $\mathcal{B}$. Assume, without loss of generality, that $\mathcal{B}$ has volume 1. Then $f_X(x)$ is 1 if $x \in \mathcal{B}$ and 0 otherwise. Given $X = x$, we choose the other endpoint $X' = x'$ according to the density λ_x. The point x' is accepted only if it lies in $\mathcal{B}$. Let I be a binary function such that for any point $q \in \mathcal{C}$, $I(q) = 1$ if $q \in \mathcal{B}$ and 0 otherwise. The conditional density of X' given X is given by

$$f_{X'|X}(x' \mid x) = \lambda_x(x')I(x')/Z_x,$$



Gaussian sampler RBB

Fig. 3. Milestones generated by the Gaussian sampler and RBB. The total number of milestones in the two cases is the same. RBB generates fewer milestones along the circular boundaries and many more in the narrow passage connecting the two circular chambers.

where $Z_x = \int_{\mathcal{C}} \lambda_x(x')I(x')\,dx'$ is a normalizing constant. To calculate π_{B} at a point $q \in \mathcal{F}$, we condition on X:

$$\pi_{\mathrm{B}}(q) = \int_{\mathcal{C}} f_{X'|X}(x' \mid x) f_X(x)\,dx. \tag{1}$$

Note that p is the midpoint of the line segment $\overline{xx'}$ and so $x' = 2q - x$. Substituting the expressions for f_X, $f_{X'|X}$, and x' into (1), we have

$$\pi_{\mathrm{B}}(q) = \int_{\mathcal{B}} \lambda_x(2q - x)I(2q - x)/Z_x\,dx. \tag{2}$$

We have chosen λ_x to be a Gaussian with its center at x and a small standard deviation. The density λ_x is large if $x' = 2q - x$ lies relatively close to x. Furthermore, the integrand in (1) is non-zero only if $I(2q - x) = 1$, *i.e.*, $x' \in \mathcal{B}$. In the neighborhood of a point q inside a narrow passage, it is more likely to find pairs of points x and x' that satisfy these two conditions, resulting in a larger value for π_{B} at q.

D. Comparison with Sampling Near Obstacle Boundaries

RBB is related to the Gaussian sampler [3]. Both use one simple geometric primitive CLEARANCE to create favorable distributions. RBB is slightly more expensive: it makes one more call to CLEARANCE in each invocation than the Gaussian sampler and rejects more samples. However, the nature of the two sample distributions generated are quite different. RBB increases the sample density in regions where short bridges can be easily constructed; the Gaussian sampler increases the sample density near obstacle boundaries. See Fig. 3 for the difference between the two sample distributions. If milestones near obstacle boundaries all improve the connectivity of roadmaps, the Gaussian sampler is preferable, as it incurs lower cost per invocation. On the other hand, obstacle boundaries may be uninteresting if they are bounding wide-open regions of $\mathcal{F}$, as samples near these boundaries do not contribute to improving the connectivity of roadmaps. Therefore, RBB gains efficiency by avoiding sampling near such boundaries. In this sense, RBB and the Gaussian sampler are complementary.

V. Combining Complementary Sampling Distributions

We have seen that RBB helps in boosting the sample density in the subset $\mathcal{P}$ of $\mathcal{F}$ occupied by narrow passages. The density

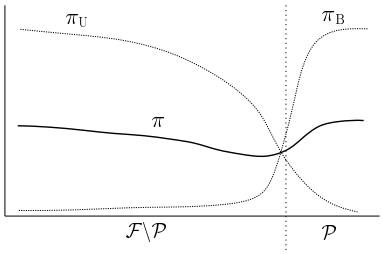


Fig. 4. The hybrid sampling distribution π. The distributions π_{B} and π_{U} perform well on $\mathcal{P}$ and $\mathcal{F}\backslash\mathcal{P}$, respectively. Combining them with suitable weights leads to good performance over the entire sampling domain $\mathcal{F}$.

π_{B} is heavily biased towards $\mathcal{P}$. At the same time, π_{B} penalizes wide-open collision-free regions: few milestones are sampled in $\mathcal{F}\backslash\mathcal{P}$. This is undesirable, because a good roadmap must cover the entire free space adequately.

Interestingly, we can make up the deficiency of π_{B} with the uniform distribution π_{U}, which samples $\mathcal{F}$ with probability proportional to the volumes of subsets in $\mathcal{F}$. For π_{U}, most milestones are sampled in $\mathcal{F}\backslash\mathcal{P}$, as it has a large volume. The two distributions complement each other naturally: π_{U} provides good coverage of $\mathcal{F}\backslash\mathcal{P}$, and π_{B} samples more densely in $\mathcal{P}$ and thus improves the connectivity of the roadmap. They are combined to produce a hybrid sampling distribution:

$$\pi = (1 - w) \cdot \pi_{\mathrm{B}} + w \cdot \pi_{\mathrm{U}}, \tag{3}$$

where w is a weight, with $0 \leq w \leq 1$. The choice of w depends on the difficulty of sampling in narrow passages and the number of milestones needed to cover $\mathcal{F}$.

One useful way of thinking about this hybrid distribution π is to divide $\mathcal{F}$ into two subsets, the set of narrow passages $\mathcal{P}$ and its complement $\mathcal{F}\backslash\mathcal{P}$. We use a different sampling strategy tailored to each subset and combine them to achieve good performance over the entire sampling domain. See Fig. 4 for an illustration. Note, however, that in our planner, the combination of two sampling strategies is achieved through weighting and not through explicit decomposition of $\mathcal{F}$.

The significance of hybrid sampling is not about putting together two sampling distributions, but rather about identifying distributions complementary in their strengths and combining them so that their individual strengths are preserved.

To implement the hybrid distribution, we can certainly generate new random points from π_{U}, but we can get some of these points "for free" by reusing the points rejected by RBB. In line 3 of Algorithm 1, RBB rejects a configuration x if CLEARANCE(x) returns TRUE, *i.e.*, x is collision-free. However, such a configuration x is exactly what is generated by π_{U}. To reduce computation time, we can save x and use it instead of generating a new one from π_{U} when needed.

VI. Implementation Issues

In this section, we describe some details for implementing our planner on rigid and articulated robots.

A. Parameterizing the Configuration Space

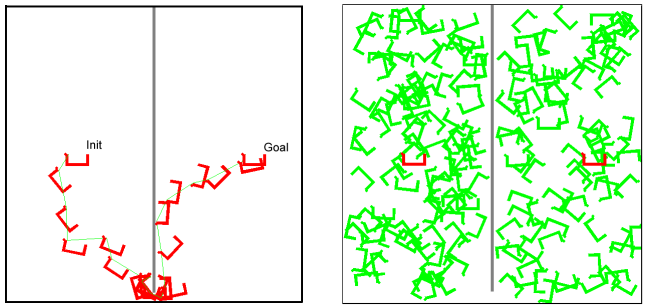
Often, each dimension of the configuration space may have a different effect on the overall motion of a robot. Consider, for example, a planar articulated robot with a free base and three links of lengths l_1, l_2, and l_3, respectively. The configuration of this robot can be represented as $(x, y, \theta_1, \theta_2, \theta_3)$, where x and y specify the position of the base, and θ_1, θ_2 and θ_3 specify the angles for the three joints in increasing order of their distance to the base along the kinematic chain. Assume that the robot is fully extended. If θ_1 changes by an angle φ, the tip of the robot moves a distance of $(l_1 + l_2 + l_3)\varphi$. If θ_2 changes by the same angle φ, the tip moves by a distance of $(l_2 + l_3)\varphi$. We must take this into account for our planner in two occasions.

First, when choosing the density function λ_x in RBB, we need a consistent measure of how restrictive the allowable motion is in the narrow passages along each dimension of the configuration space $\mathcal{C}$. Our solution is to rescale $\mathcal{C}$. For each configuration space coordinate q_i, let d_i be the maximum distance traveled by any point on the robot when the robot moves between any two configurations that have identical coordinates in all dimensions except q_i [32]. We rescale the range of q_i to $[0, d_i]$. This method of rescaling is quite general and can be applied to any translational or rotational dof of a robot. In our planar articulated robot example, for translational dofs, the scaling factors are 1, and rescaling has no net effect. For rotational dofs, we have, for example, $d_2 = (l_2 + l_3)2\pi$ for the joint angle θ_2, and we can scale other joint angles similarly. For rigid robots in 3-D, rotational dofs can be represented with either Euler angles or quaternions. If Euler angles are used, the scaling factor for each Euler angle is calculated exactly the same way as that for the joint angle of articulated robots: $d_i = 2\pi L_i$, where L_i is the maximum distance of any point on the robot to the axis of rotation for the corresponding Euler angle q_i. If quaternions are used, the calculation is slightly more involved, but the principle is the same.

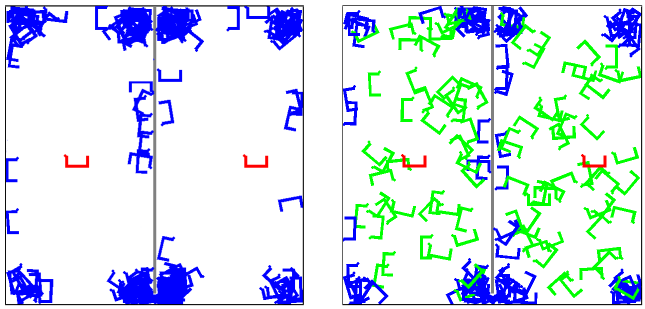
We need to consider this scaling issue again, when adding a new milestone q to the roadmap. The planner tries to connect q with existing milestones within a certain distance. So we need a suitably-defined distance metric on $\mathcal{C}$. Ideally the metric has the property that for any two free configurations q and q', the greater the distance between q and q' is, the more likely that the robot encounters an obstacle when following a straight-line path from q to q'. Again we have to rescale $\mathcal{C}$ so that each dimension of $\mathcal{C}$ has a similar effect on the overall motion of the robot. After rescaling, we can simply define the metric on $\mathcal{C}$ to be the Euclidean distance between two configurations. This scaling heuristic is often used in motion planning (see, *e.g.*, [7]). Other heuristics can also be used. For instance, instead of using the maximum distance traveled by any point on the robot as a measure of the "distance" between two configurations, we can use the volume swept out by the robot.

B. Parameters for the Sampling Distribution

Two parameters are needed to fix the hybrid sampling distribution π. The first parameter fixes the density function λ_x for RBB. As we have discussed earlier, λ_x is a product of independent Gaussians, with a small standard deviation σ to bias towards sampling short bridges. The parameter σ depends on the width of narrow passages to be captured and may affect the planner's performance. In practice, we can estimate the



(*a*) Query configurations and a path found. (*b*) Uniform sampling.



(*c*) RBB. (*d*) Hybrid sampling.

Fig. 5. A planar rigid-body robot maneuvering through a small opening between two large chambers.

value of σ by analyzing the geometry of the robot and the obstacles in the environment. For example, if the smallest passage in the workspace has width W and a rigid robot, which translates and rotates, has maximum radius R, then σ can be set to be proportional to W/R.

The second parameter is the weight for combining π_{B} and π_{U}. For our experiments, we assume no prior knowledge of the environment. We do not bias towards either π_{B} or π_{U} and use a relative weight of 1:1.

The best parameters settings are clearly problem-dependent, and it is difficult to choose them in advance. A promising approach is to use a set of RBBs with different σ values and adjust the weight for each RBB adaptively through online learning [28]. This way, the best parameter values can be identified automatically in an adaptive way.

VII. Experiments

We implemented two versions of our planner, one version in Java for robots in 2-D environments and one version in C++ for robots in 3-D environments. To examine the planner's performance, we hand-crafted several difficult environments and tested our implementations extensively. The 2-D test environments are described below:

- Fig. 5*a*: We have a rigid-body robot, and the query asks the robot to go from one large empty chamber to another through a small opening near the lower middle of the figure. This is a case where we expect the hybrid sampling strategy to work well, because the bridge test eliminates many sampled configurations that lie in the middle of wide-open space and are unlikely to improve the coverage or connectivity of the roadmap. Figures 5*b*–5*d* show the roadmaps generated by uniform sampling, pure RBB, and hybrid sampling, respectively. Each sampling strategy uses the same number of milestones. The roadmap constructed by uniform sampling covers the free space well, but wastes many milestones in the two chambers and does not put any milestone near the small opening. Note that in these figures, the milestones are projected from the 3-dimensional configuration space to the 2-dimensional workspace for drawing. Two milestones appear close to each other in the figures may be far away from each other in the configuration space because of different orientations, and therefore there may not be a collision-free straight-line path that connects them. Pure RBB puts a large number of milestones near places where it can successfully build "bridges", including the small opening between chambers. This enables it to capture the connectivity of the free space well. However, the roadmap constructed by RBB has poor coverage: very few milestones are in the two chambers to cover the free space. This does not cause a big problem here, because the two chambers in this example are convex, and every point inside a chamber can cover a large portion of the chamber. However, in general, more milestones are needed, if the chambers have more complex geometry. Hybrid sampling combines the strengths of uniform sampling and RBB to build a roadmap that has good coverage and captures the connectivity of the free space well.
- Fig. 6: According to our analysis in Section III, sampling with the bridge test works well, only if the resulting roadmap contains much fewer milestones than those generated by other sampling strategies. We constructed this example so that the bridge test does not have such an advantage. This planar environment for a point robot contains a long path that has almost equal width everywhere. So almost every sampled configuration passes the bridge test. The roadmap constructed with the bridge test would have roughly the same size as that constructed by uniform sampling or Gaussian sampling.
- Fig. 7: A rigid-segment robot enters a narrow corridor, reorients in a small circular room, and exits another narrow corridor. This environment is an example of connected narrow passages in different orientations. When the robot is inside one of the two corridors, it can only translate along the direction of the corridor. Movements in the orthogonal directions are very restricted: it cannot translate sidewise or rotate. When the robot is inside the small circular room in the middle, the rotational movement is unrestricted, but the translational movement is very restricted, as the diameter of the room is only slightly more than the length of the robot. In the 3-dimensional configuration space, the two corridors and the round room in the middle map to three connected narrow passages in different orientations.
- Fig. 8: In this example, we have a T-shaped robot with two parts, a "torso" and a "shoulder", connected by a joint. The planar environment contains a long and narrow corridor with two turns.

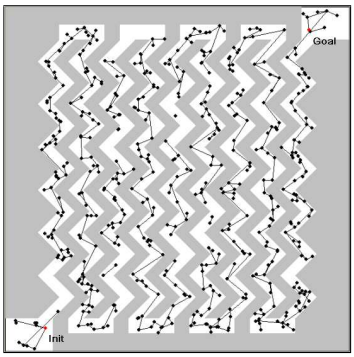


Fig. 6. A partial roadmap built by RBB for a point robot in the plane.

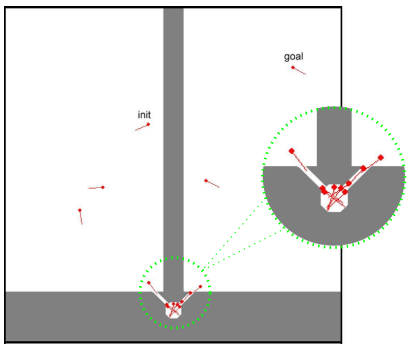


Fig. 7. A path for a rigid segment translating and rotating in a narrow region.

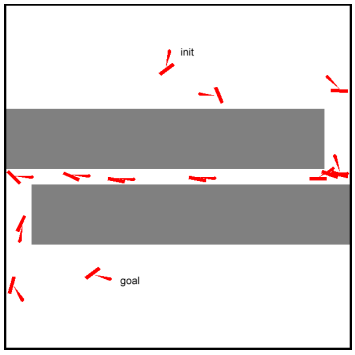


Fig. 8. A path for a T-shaped robot moving through a long, narrow corridor.

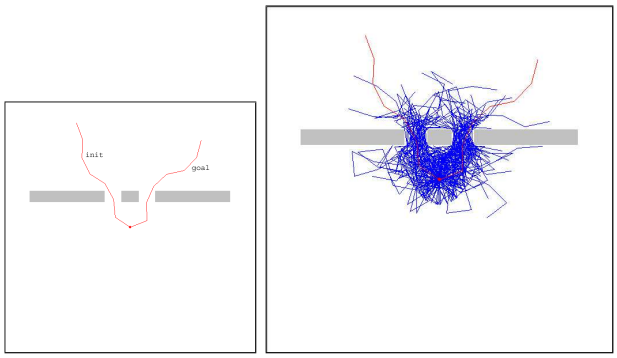


Fig. 9. A 7-dof articulated robot with a fixed base. The left figure shows the query configurations. The right figure shows those milestones generated by RBB.

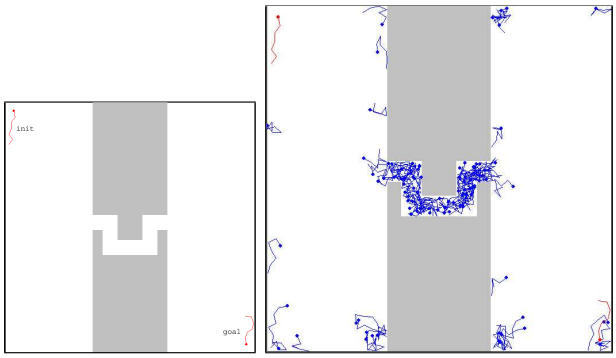


Fig. 10. An 8-dof articulated robot with a mobile base. The left figure shows the query configurations. The right figure shows those milestones generated by RBB.

- Fig. 9: This environment contains a seven-dof articulated robot with a fixed base. At the initial configuration, the robot is trapped inside a narrow opening. Joint angles near the robot's base have very limited range of motion. Joint angles near the robot's tip can move relatively freely. As the robot performs difficult maneuvers to pull out of the narrow opening, an increasing number of joints have their motion restricted, until the robot pulls out completely and all the joints can move freely. The robot then has to insert itself into the other narrow opening to reach the goal configuration. The sequence of events are similar, but occur in reverse order.
- Fig. 10: This environment contains a relatively short corridor with two turns. Each milestone in the corridor covers only a small portion of the free space. The robot is an articulated robot with six links and a mobile base, eight dofs in total.

We tested these environments with the Java implementation of our planner. For each test environment, we rescaled the configuration space $\mathcal{C}$ to $[0, d_1] \times [0, d_2] \times \cdots$, as described in Section VI. Since all the environments are bounded, we applied an additional uniform scaling $1/d$, where $d = \max_i d_i$, so that $\mathcal{C}$ fits within a unit hyper-cube, in order to simplify the implementation.

The relative weight for combining the two components π_{U} and π_{B} of the hybrid sampling distribution was 1:1, *i.e.*, equal weights for the two components. We also systematically varied the standard deviation σ of the Gaussian for the bridge test between $1/128$ and $1/16$ to choose the best setting. It turned out that $\sigma = 1/32$ worked well in all the experiments.

To add a new milestone q to the roadmap, the planner tries to connect q with an existing milestone q', only if (i) q and q' are in different connected components, (ii) the distance between q and q' is smaller than a threshold D, and (iii) q' is a K-nearest neighbor of q. This method of adding nodes, called nearest-K, has been used in earlier work for comparing sampling strategies [25], [26] and shown robust performance in extensive experiments. Based on earlier experimental results (see, *e.g.*, [25], [26]) and our own experiences, we have chosen D to be 0.25 and K to be 20. We have intentionally chosen slightly lower values so that our planner does not get unfair advantages.

We also ran the same experiments with two other sampling strategies, the uniform sampler and pure RBB (without mixing with uniform sampling). The uniform sampler is used mainly as a way to calibrate the difficulty of the queries, and RBB is used to examine the benefit of hybrid sampling. For RBB, the standard deviation σ of the Gaussian is set to $1/8$. Without the help from the uniform sampler, we cannot make σ too small. Otherwise, it would reduce the coverage of the roadmap and adversely affect the performance of the pure RBB.

For comparison, we also experimented with the Gaussian sampler and visibility-based PRM (Vis-PRM), with optimized

TABLE I
THE PERFORMANCE OF VARIOUS SAMPLING STRATEGIES.

Env. (Fig.)	dofs	Sampler	N_{mil}	Time (sec.) avg.	std.
5	3	Hybrid	302	3.9	2.2
		RBB	4,555	30.0	48.1
		Uniform	25,607	382.4	350.6
		Visibility	83	120.3	96.5
		Gaussian	1,165	5.6	3.2
7	3	Hybrid	900	7.4	2.8
		RBB	5,956	34.7	25.9
		Uniform	47,064	995.0	234.7
		Visibility	108	235.9	106.8
		Gaussian	2,276	8.4	3.9
8	4	Hybrid	916	7.1	2.9
		RBB	3,422	17.1	6.3
		Uniform	23,152	354.0	204.3
		Visibility	209	148.8	45.3
		Gaussian	1,976	9.9	3.8
9	7	Hybrid	1,307	111.3	42.1
		RBB	2,915	231.1	85.5
		Uniform	22,465	1012.3	380.6
		Visibility	826	790.6	288.5
		Gaussian	1,619	146.5	54.3
10	8	Hybrid	1,004	41.4	13.3
		RBB	4,155	98.2	26.2
		Uniform	35,822	1803.3	783.2
		Visibility	620	698.1	173.8
		Gaussian	3,255	92.2	27.6
6	2	Hybrid	1,317	25.6	6.3
		RBB	1,390	26.4	5.5
		Uniform	1,358	23.4	5.4
		Visibility	206	42.6	11.5
		Gaussian	1,291	23.5	4.3

parameter settings. In particular, for the Gaussian sampler, we varied the standard deviation of the Gaussian, using five different settings between $1/512$ and $1/16$, and chose the best one for each environment.

We ran each sampling strategy 30 times independently for each test environment, and terminated the planner as soon as a path was found between query configurations. The average number of milestones in the final roadmap, N_{mil}, and the average running time are shown in Table I. The statistics were gathered from the Java implementation of our planner on a Linux workstation with a 2.8 GHz Pentium 4 processor.

Table I shows consistent results in these tests with different robots and environments. Hybrid sampling usually outperforms both the uniform sampler and pure RBB. Its roadmap is much smaller (see column 4), and the total running time is also shorter. The reduction in running time is not proportional to the reduction in roadmap size, because hybrid sampling pays a higher cost to obtain a milestone than the simpler uniform sampling. However, the smaller roadmap size requires much fewer tests to check collision-free connections between the milestones in the roadmap (see Table II), which are the dominant factor in the total running time. So hybrid sampling is able to achieve good overall performance. This basically confirms our intuition on the benefit of using the bridge test, as described in Section III.

For hybrid sampling, the standard deviation σ used for the bridge test does affect the performance, sometimes by as much as 50%. Nevertheless, hybrid sampling performs better than uniform sampling and RBB in all the environments with narrow passages, as long as a reasonable, not necessarily the best, value for σ is chosen. This shows the benefit of hybrid sampling even if we do not know the best value for σ.

TABLE II
A BREAKDOWN OF THE TOTAL RUNNING TIME INTO COMPONENTS DEFINED IN SECTION III FOR THE EXAMPLE IN FIG. 10. T_{mil} AND T_{con} ARE MEASURED IN THE NUMBER OF CALLS TO CLEARANCE.

Sampler	N_{mil}	T_{mil}	N_{con}	T_{con}	Time (sec.)
Hybrid	1,004	113.5	7,180	43.3	41.4
RBB	4,155	33.4	11,127	59.0	98.2
Uniform	35,822	1.8	50,315	66.2	1,803.3
Visibility	620	10825.2	4,955	93.9	698.1
Gaussian	3,255	18.5	15,699	46.4	92.2

In most of these experiments, RBB alone without uniform sampling does not perform as well as hybrid sampling, because these environments contain both narrow passages and wide-open free space. Without uniform sampling, we have to set σ for the bridge test to a relatively large value in order to improve the coverage of the roadmap. As a result, the ability of the bridge test to identify narrow passages is reduced, leading to worse performance.

For Vis-PRM and the Gaussian sampler, as well as our hybrid sampling strategy, they all try to reduce the size of the roadmap and improve computational efficiency by filtering out milestones that are not useful. Yet they differ in the filtering cost (see Table II, column 3 for an example). To certify a useful milestone, Vis-PRM must perform several tests to check collision-free connection between milestones, a very expensive operation. The cost of filtering for both Gaussian sampling and hybrid sampling is much cheaper. Gaussian sampling takes two calls to CLEARANCE to certify a useful milestone, and RBB takes three calls. As a result, although Vis-PRM produces very small roadmaps (see Table II, column 2) and clearly improves over uniform sampling, its performance is weaker than the Gaussian sampler and the hybrid sampling strategy using RBB. This is expected in light of our discussion in Section III, and is why we suggest that T_{mil} must remain small for the filtering to be effective. Gaussian sampling assumes that useful milestones lie close to obstacle boundaries. When this assumption holds, it has an advantage over hybrid sampling (see the example in Fig. 6). When the assumption fails, it may be substantially slower than hybrid sampling, especially for robots with many dofs, as the example in Fig. 10 shows.

The first five environments in Table I share a common characteristic: they all contain narrow passages connecting regions that allow relative unrestricted movement. This gives an advantage to sampling strategies designed specifically for narrow passages, such as the hybrid sampling strategy or RBB. The environment in Fig. 6 is different. It contains a long path that has almost equal width everywhere. The resulting roadmaps all have roughly the same size. So the uniform sampler, which has low cost for sampling a milestone, performs better. However, even in this case, hybrid sampling performs only 9% worse than uniform sampling.

We also tested the C++ implementation of our planner in 3-D environments. For each environment, we manually

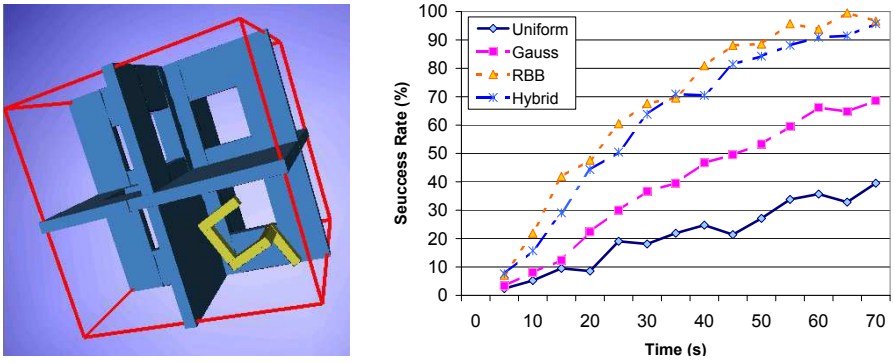


Fig. 11. A rigid body translating and rotating freely in a 3-D environment.

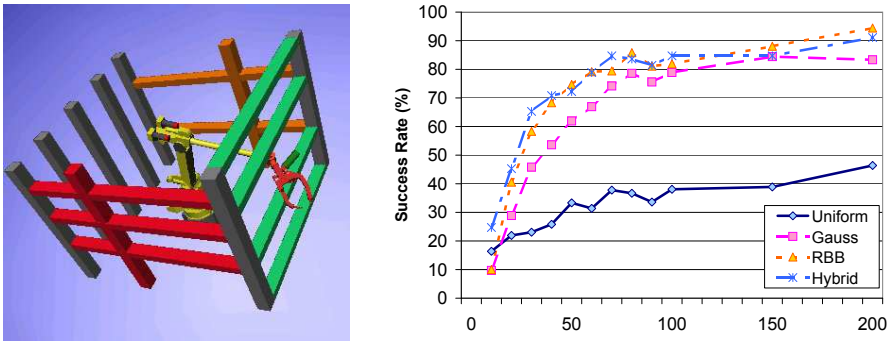


Fig. 12. A 6-dof robot manipulator.

specified several queries so that solving these queries indicates that a roadmap captures the connectivity of the free space well. Again each test was repeated 30 times independently. For each test environment below, we show a graph that plots the percentage of queries that a planner can answer correctly as the running time increases.

- Fig. 11: This test uses a rigid-body robot translating and rotating freely in a 3-D environment. The space is divided into eight chambers by walls with holes. We specified query configurations in all the chambers and ran the planners until all possible connections are established among the query configurations. The configuration space here consists of many narrow passages, with a moderate amount of free space that allows unrestricted movement. Pure RBB turns out to have the best performance, because it focuses on sampling in narrow passages. The performance of the hybrid sampling strategy is close.
- Fig. 12: Here we have a six-dof robot manipulator arm. Horizontal and vertical bars are set up around the robot to make its movement difficult. We specified 12 queries. Each query requires the robot to move its end-effector from one opening between the bars to another. To answer a query, the robot must pull its end-effector out of a narrow opening, move in relatively open free space, and reinsert the end-effector into another narrow opening. In this test, the hybrid sampling strategy, RBB, and the Gaussian sampler have similar performance, all significantly better than uniform sampling.

VIII. Discussion

Compared with the idea of sampling near obstacle boundaries, the bridge test gains efficiency by filtering out those samples near uninteresting obstacle boundaries, but it is not perfect. Some uninteresting samples near corners can pass the bridge test, because near the tip of a corner, it is easy to build short bridges. Fig. 5*c* shows that RBB generated a number of milestones near the corners of $\mathcal{F}$, and many of these milestones may be unhelpful. The bridge test generates false positives in this case, because it is a test of *local* geometry. As Fig. 13 shows, if we only have information within a small neighborhood, we cannot tell the difference between a narrow passage and a narrow dead-end. Despite this problem, our experiments show that the benefits gained by sampling in narrow passages usually outweigh the computation time wasted in sampling near corners (see Section VII).

We can try to reduce the false positives near corners by taking additional samples. For example, the *orthogonal test* picks at random an additional segment s' through the configuration q such that the segment s' is orthogonal to the bridge s. We accept q as a new milestone, if the endpoints of s' are both in collision or both free, in addition to the normal conditions on s. This helps to reduce the false positives, because such samples often have one endpoint in collision and one endpoint free. However, the orthogonal test is more expensive: it takes five calls to CLEARANCE per test in the worst case, while the original bridge test takes only three.

What is in common among the orthogonal test, the bridge test, and the Gaussian sampler is that they all try to reconstruct

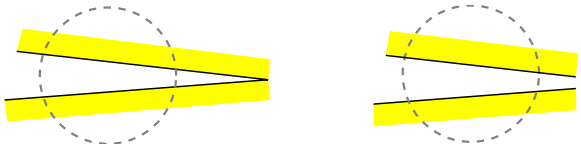


Fig. 13. A narrow passage and a narrow dead-end. Within the small neighborhood marked by the dashed circles, there is not enough information to differentiate these two cases.

the local geometry of the configuration space by taking a small number of samples. We can think of other tests based on local sampling as well. The more samples we take, the more likely that we can reconstruct the geometry accurately, at least in principle; at the same time, we have to pay a higher cost for sampling. In the extreme case, we can also check the collision-free connection between two sampled points, as Vis-PRM does. This yields very accurate information about the connectivity of the local free space, but it is very expensive. Our experiments indicate that the filtering cost should remain low in order for it to be effective.

IX. Conclusion and Future Work

We have presented a hybrid sampling strategy to address the narrow passage problem for PRM planning. A key ingredient of the new sampling strategy is the bridge test, which boosts the sample density in narrow passages. The bridge test can be viewed as a filter that rejects the milestones that are unlikely to improve the connectivity of a roadmap and thus saves the computation time by avoiding the expensive tests needed to connect these milestones in the roadmap. The bridge test is purely local and can be implemented efficiently in high-dimensional configuration spaces. By combining the bridge test with uniform sampling, we construct a hybrid sampling strategy that generates small roadmaps that cover the free space well and have good connectivity. Our experiments on rigid and articulated robots in 2-D and 3-D environments show that our planner was able to reliably capture the connectivity of free spaces with difficult narrow passages.

There are two main issues that we are interested in exploring further in the future.

First, in our current hybrid sampling strategy, the weight for combining the bridge test and the uniform sampler is set manually, and the best choice is clearly problem dependent. A promising approach is to adjust the weights adaptively through on-line learning. It would also be interesting exploit the possibly complementary strengths of RBB and the Gausssian sampler and combine them as well as the uniform sampler in a hybrid sampling framework. Work is currently underway in this direction [28].

Second, viewing the bridge test as a filter allows us to combine it with other sampling strategies. For example, MAPRM [22] samples points on the medial axis of $\mathcal{F}$. Since the milestones on the medial axis of the wide-open regions of $\mathcal{F}$ tend to have good coverage of $\mathcal{F}$, removing some of them does not affect the connectivity of the roadmap. On the other hand, milestones on the medial axis of narrow passages are more critical, as they have much more limited visibility. Removing any of them may disconnect the roadmap. Therefore, after generating milestones using MAPRM, we can apply the bridge test to the milestones. Each milestone that passes the bridge test is retained; each one that fails is retained with a certain probability. This way, we can get a smaller roadmap without affecting its quality.

Acknowledgement

We thank Jean-Claude Latombe for many insightful comments and anonymous reviewers for suggestions that have helped strengthen the paper.

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 3rd Workshop on Algorithmic Foundations of Robotics*, 1998, pp. 155–168.

[2] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, 2000, pp. 521–528.

[3] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, 1999, pp. 1018–1023.

[4] L. K. Dale and N. M. Amato, "Probabilistic roadmaps—putting it all together," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, 2001, pp. 1940–1947.

[5] M. Foskey, M. Garber, M. C. Lin, and D. Manocha, "A Voronoi-based hybrid motion planner," in *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 55–60.

[6] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *International Journal of Computational Geometry & Applications*, vol. 9, no. 4 &5, pp. 495–512, 1999.

[7] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, 1996.

[8] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 278–400, 2001.

[9] S. R. Lindemann and S. M. LaValle, "Incremental low-discrepancy lattice methods for motion planning," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, 2003, pp. 2920–2927.

[10] C. Nissoux, T. Siméon, and J.-P. Laumond, "Visibility based probabilistic roadmaps," in *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1999, pp. 1316–1321.

[11] N. M. Amato, K. A. Dill, and G. Song, "Using motion planning to map protein folding landscapes and analyze folding kinetics of known native structures," in *Proceedings of the 6th Annual International Conference on Computational Biology*, 2002, pp. 2–11.

[12] M. S. Apaydin, D. L. Brutlag, C. Guestrin, D. Hsu, and J.-C. Latombe, "Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion," in *Proceedings of the 6th Annual International Conference on Computational Biology*, 2002, pp. 12–21.

[13] H. Chang and T.-Y. Li, "Assembly maintainability study with motion planning," in *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, 1995, pp. 1012–1019.

[14] T. Siméon, J.-P. Laumond, C. Geem, and J. Cortes, "Computer aided motion: Move3D within MOLOG," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, 2001, pp. 1494–1499.

[15] A. P. Singh, J.-C. Latombe, and D. L. Brutlag, "A motion planning approach to flexible ligand binding," in *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, 1999, pp. 252–261.

[16] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, 1979, pp. 421–427.

[17] J. H. Reif and Z. Sun, "On frictional mechanical systems and their computational power," *SIAM Journal on Computing*, vol. 32, no. 6, pp. 1449–1474, 2003.
[18] J. Sellen, "Lower bounds for geometrical and physical problems," *SIAM Journal on Computing*, vol. 25, no. 6, pp. 1231–1253, 1996.
[19] J. Barraquand, L. E. Kavraki, J.-C. Latombe, T. Li, R. Motwani, and P. Raghavan, "A random sampling scheme for path planning," *International Journal of Robotics Research*, vol. 16, no. 6, pp. 759–774, 1997.
[20] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwari, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *Proceedings of the 3rd Workshop on Algorithmic Foundations of Robotics*, 1998, pp. 141–153.
[21] A. D. Collins, P. K. Agarwal, and J. L. Harer, "HPRM: A hierarchical PRM," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, 2003, pp. 4433–4438.
[22] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "Motion planning for a rigid body using random networks on the medial axis of the free space," in *Proceedings of the 15th Annual ACM Symposium on Computational Geometry*, 1999, pp. 173–180.
[23] C. Holleman and L. E. Kavraki, "A framework for using the workspace medial axis in PRM planners," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, 2000, pp. 1408–1413.
[24] Y. Yang and O. Brock, "Adapting the sampling distribution in PRM planners based on an approximated medial axis," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, 2004.
[25] R. Geraerts and M. H. Overmars, "A comparative study of probabilistic roadmap planners," in *Algorithmic Foundations of Robotics V*. Springer, 2002, pp. 43–59.
[26] ——, "Sampling techniques for probabilistic roadmap planners," in *Proceedings of the 8th Conference on Intelligent Autonomous Systems*, 2004, pp. 600–609.
[27] D. Hsu and Z. Sun, "Adaptively combining multiple sampling strategies for probabilistic roadmap planning," in *Proceedings of the IEEE Conference on Robotics, Automation, and Mechatronics*, 2004, pp. 774–779.
[28] D. Hsu, G. Sánchez-Ante, and Z. Sun, "Hybrid PRM sampling with a cost-sensitive adaptive strategy," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 3885–3891.
[29] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato, "A machine learning approach for feature-sensitive motion planning," in *Proceedings of the 6th Workshop on Algorithmic Foundations of Robotics*, 2004.
[30] S. Quinlan, "Efficient distance computation between non-convex objects," in *Proceedings of the 1994 International Conference on Robotics and Automation*, 1994, pp. 3324–3330.
[31] S. Gottschalk, M. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 1996, pp. 171–180.
[32] J.-C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.