UNIFORM SAMPLING FRAMEWORK FOR SAMPLING BASED MOTION PLANNING AND ITS APPLICATIONS TO ROBOTICS AND PROTEIN LIGAND BINDING

A Dissertation

by

HSIN YI YEH

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Nancy M. Amato
Committee Members,	J. Martin Scholtz
	Dezhen Song
	Tiffani L. Williams
Head of Department,	Dilma Da Silva

May 2016

Major Subject: Computer Science

ABSTRACT

Sampling-based motion planning aims to find a valid path from a start to a goal by sampling in the planning space. Planning on surfaces is an important problem in many research problems, including traditional robotics and computational biology. It is also a difficult research question to plan on surfaces as the surface is only a small subspace of the entire planning space. For example, robots are currently widely used for product assembly. Contact between the robot manipulator and the product are required to assemble each piece precisely. The configurations in which the robot fingers are in contact with the object form a surface in the planning space. However, these configurations are only a small proportion of all possible robot configurations. Several sampling-based motion planners aim to bias sampling to specific surfaces, such as $\mathcal{C}_{obst}$ surfaces, as needed for tasks requiring contact, or along the medial axis, which maximizes clearance. While some of these methods work well in practice, none of them are able to provide any information regarding the distribution of the samples they generate. It would be interesting and useful to know, for example, that a particular surface has been sampled uniformly so that one could argue regarding the probability of finding a path on that surface. Unfortunately, despite great interest for nearly two decades, it has remained an open problem to develop a method for sampling on such surfaces that can provide any information regarding the distribution of the resulting samples.

Our research focuses on solving this open problem and introduces a framework that is guaranteed to uniformly sample any surface in $\mathcal{C}_{space}$. Instead of explicitly constructing the target surfaces, which is generally intractable, our uniform sam-

pling framework only requires detecting intersections between a line segment and the target surface, which can often be done efficiently. Intuitively, since we uniformly distribute the line segments, the intersections between the segments and the surfaces will also be uniformly distributed. We present two particular instances of the framework: Uniform Obstacle-based PRM (UOBPRM) that uniformly samples $\mathcal{C}_{obst}$ surfaces, and Uniform Medial-Axis PRM (UMAPRM) that uniformly samples the $\mathcal{C}_{space}$ medial axis. We provide a theoretical analysis for this framework that establishes uniformity and probabilistic completeness and also the probability of sampling in narrow passages. We show applications of this uniform sampling framework in robotics (both UOBPRM and UMAPRM) and in biology (UOBPRM). We are able to solve some difficult motion planning problems more efficiently than other sampling methods, including PRM, OBPRM, Gaussian PRM, Bridge Test PRM, and MAPRM. Moreover, we show that UOBPRM and UMAPRM have similar computational overhead as other approaches. UOBPRM is used to study the ligand binding affinity ranking problem in computational biology. Our experimental results show that UOBPRM is a potential technique to rank ligand binding affinity which can be further applied as a cost-saving tool for pharmaceutical companies to narrow the search for drug candidates.

DEDICATION

To my parents: you are always my strong foundation

To my brother, Shu-Hao: you are my constant support

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dr. Nancy Amato, for her support and encouragement. I deeply appreciate all her guidance and care through the years.

I would also like to thank my committee members, Dr. J. Martin Scholtz, Dr. Dezhen Song, and Dr. Tiffani L. Williams, for their support and feedback. I appreciate they committed their time to help me grow as a researcher.

I also thank all the collaborators that I have worked with over the years, both on this work and on other research projects: Jory Denny, Chinwe Ekenna, Mukulika Ghosh, Aaron Lindsey, Dr. Lydia Tapia, Dr. Shawna Thomas, and Chih-Peng Wu. I am especially thankful to Shawna for her guidance over the years. Thank you for your extreme patience as I explored research problems in motion planning and computational biology. I am also grateful for working with Chinwe and Mukulika through most of my graduate career. Since we've shared an office together, we collaborated not only on research but also the service in AWICS. We have accomplished a lot by working together.

Thank you to all the Parasol members, both former and current.

Thanks to Texas A&M University Diversity Fellowship who have supported my graduate career. I would also like to thank all the conferences and workshops that provided travel grants during my graduate career. These include funding to attend the Grace Hopper Celebration of Women in Computing Conference, the IEEE/RSJ International Conference on Intelligent Robots and Systems, the IEEE International Conference on Robotics and Automation, and the ACM Conference on Bioinformatics, Computational Biology, and Health Informatics. It helped to broaden my view.

Finally, I would like to thank my family who always support me and stay with me through my graduate career. My parents taught me the value of the hard work. Their encouragement helped me throughout this long journey. My brother, Shu-Hao, has been the one who always reminds me the joy of learning new things. Thank you for always providing me the courage of facing new challenges.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

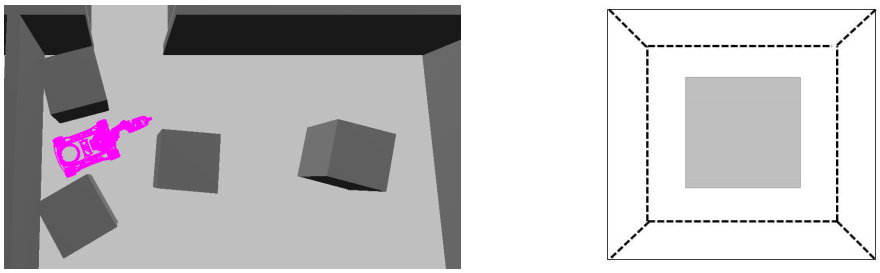
1. INTRODUCTION

The motion planning problem is to find a valid (e.g., collision free) trajectory for a movable object (robot) from a start position to a goal position. Motion planning has been studied extensively and has various applications such as robotics [11, 34, 42], computer animation [41, 6], computer-aided design (CAD) [18, 59], and computational biology [57, 4, 1, 63]. Motion planning is very difficult, known to be intractable for even very simple problems [55, 5]. Consequently, randomized sampling-based planners have become the state-of-the-art methods for planning [40, 34, 47, 46, 2, 3, 13, 32, 68, 49, 30, 72, 50]. These planners build graphs [40] or trees [34, 47] that approximate the topology of the planning space and encode representative feasible trajectories.

The focus of this dissertation is to study motion planning on surfaces. This is a challenging and important problem with applications in many research areas, including traditional robotics and computational biology. The dimension of a surface is one less than the dimension of the planning space, hence sampling on the surface can be difficult because the surface is a relatively small space compared to the entire planning space. For example, robots are widely used for product assembly nowadays. In order to precisely assemble each piece, contact between the robot manipulator and the product are required. The configurations in which the robot fingers are in contact with the object form a surface in the planning space. However, these configurations are only a small percentage of all possible robot configurations. Therefore, it is very difficult to obtain such configurations with sampling-based planners.

Many problems in robotics require robots to operate in cluttered environments with narrow passages. For example, as shown in Figure 1.1(a), a KUKA youBot [43]

tries to get to the next room by passing through a narrow passage. As the robot is very close to the surrounding obstacles, even small movements can cause the robot to collide with obstacles. The medial axis is a surface that maximizes clearance and hence, planning paths on medial axis surfaces is desirable for many applications (Figure 1.1(b)). Another example is when the task requires contact between the robot and an object; since the robot configurations that are in contact with the object form a surface in the planning space, this is a planning problem in which paths must be found on this surface.

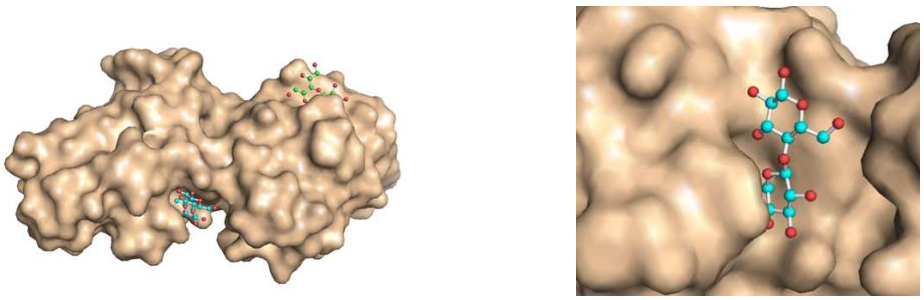


(a) (b)

Figure 1.1: (a) A KUKA youBot [43] needs to pass through the narrow passage caused by the surrounding obstacles in order to approach to the entrance to the next room. (b) Planning on the medial axis in the narrow passage provides a high clearance path since the medial axis is a surface that is equidistant to two or more obstacles.

Planning on surfaces also has applications in computational biology. One example is the ligand binding problem. Protein-ligand interaction is essential to understand many biological mechanisms. The efficiency of a drug (ligand) molecule is determined by its ability to find a specific position and orientation on the protein surface, more specifically, on the surface in the binding pocket (binding site, see Figure 1.2(b)). This contact (called binding) on the protein surface can either activate or inhibit

some biochemical effects. For example, the binding between insulin and the insulin receptor will trigger some intracellular insulin effects, such as fat metabolism and glucose uptake [56]. Planning the ligand motion as it approaches and then attaches to the protein surface is a motion planning problem requiring contact. Sampling-based motion planning can also be used in ligand binding site prediction, which helps to predict where on the protein surface the ligand may form contact and trigger biochemical effects [71, 70, 67, 15, 16]. Sampling-based motion planning can be used to map the planning space of the protein, which can be utilized to predict binding sites (Figure 1.2(a)).



(a) (b)

Figure 1.2: (a) Binding between the protein 1UYX and two ligands. (b) The binding pocket (binding site) is a small region on the protein surface where the ligand can form chemical bonds to cause some biochemical effects. The images were generated with PyMOL [25].

Many motion planning methods have been specialized for planning on or near surfaces, such as near obstacle surfaces [2, 13, 32] or along medial axis surfaces [68, 49], to improve performance or to find paths with desirable properties (e.g., high clearance). OBPRM [2] (Figure 1.3(d)) was the first method targeting sampling on obstacle surfaces, and then Gaussian PRM [13] (Figure 1.3(e)) and Bridge Test

PRM [32] (Figure 1.3(f)) were proposed to generate samples on obstacle surfaces or in narrow passages, respectively. MAPRM [68] (Figure 1.3(g)) biases sampling towards medial axis surfaces. While some of these methods work well in practice, none of them is able to provide any information regarding the distribution of the samples it generates. It would be interesting and useful to know, for example, that a particular surface has been sampled uniformly so that one could argue regarding the probability of finding a path on that surface. Unfortunately, despite great interest for nearly two decades, it has remained an open problem to develop a method for sampling on such surfaces that can provide any information regarding the distribution of the resulting samples.

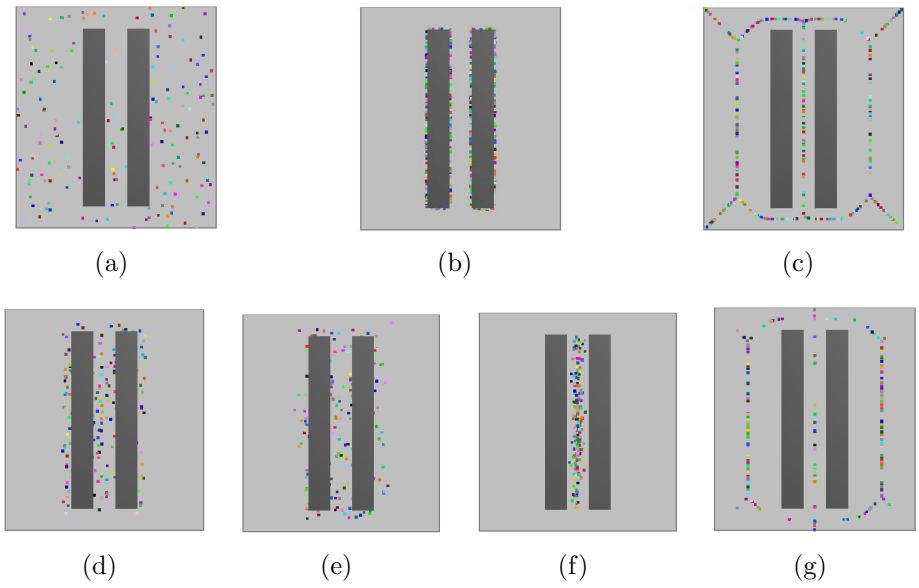


Figure 1.3: Nodes generated by (a) PRM [40], (b) UOBPRM [22], (c) UMAPRM [73], (d) OBPRM [2], (e) Gaussian PRM [13], (f) Bridge Test PRM [32], and (g) MAPRM [68] in a simple 2D environment containing two parallel obstacles. PRM, UOBPRM, and UMAPRM can guarantee uniformly distributed samples on their respective targeted surfaces.

1.1 Research Contribution

In this dissertation, we present a solution to a long standing open problem and develop a general method that can uniformly sample surfaces. Instead of explicitly constructing the target surfaces, which is generally intractable, our uniform sampling framework only requires detecting intersections between a line segment and the target surface, which can often be done efficiently. Intuitively, since we uniformly distribute the line segments, the intersections between the segments and the surfaces will also be uniformly distributed.

We use the uniform sampling framework in sampling-based motion planning to study some important surfaces in the planning space. Uniform Obstacle-based PRM (UOBPRM) [22] is the first example of our framework which samples obstacle surfaces uniformly (Figure 1.3(b)). Uniform Medial-Axis PRM (UMAPRM) [73] is another example whose target surfaces are the medial axis of $\mathcal{C}_{space}$ (Figure 1.3(c)). We show that this framework generates configurations uniformly distributed on the target surfaces of $\mathcal{C}_{space}$ (all possible robot placements), both experimentally and theoretically. We prove that the probability of sampling the target surfaces is proportional to their surface area, which leads to important observations regarding the probability of generating samples in narrow passages. Next, we prove that the uniform sampling framework is probabilistically complete. We also formalize the relationship between UOBPRM and Gaussian PRM [13] and show that Gaussian PRM is a special case of UOBPRM with particular parameter settings.

We show applications of this uniform sampling framework in robotics (both UOBPRM and UMAPRM) and in biology (UOBPRM). We are able to solve some difficult motion planning problems more efficiently than other sampling methods, including PRM [40], OBPRM [2], Gaussian PRM [13], Bridge Test PRM [32], and

MAPRM [68]. Our results show that both UOBPRM and UMAPRM have negligible computational overhead over other sampling techniques and UMAPRM can solve problems that others could not (e.g., a bug trap environment). We illustrate how UOBPRM can be used to study the ligand binding affinity ranking problem by generating uniformly distributed ligand samples on the target protein's surfaces. Experiments with several target proteins using two different experimental measures for binding affinity show that UOBPRM can potentially rank binding affinities for different ligands.

In summary, our contributions include a uniform sampling framework that uniformly generates configurations on surfaces and its instances for different target surfaces (e.g., $\mathcal{C}_{obst}$ surfaces and medial axis surfaces). We evaluate the framework on a variety of applications. More specifically, our contributions are as follows:

- We present a general uniform sampling framework that distributes samples uniformly on surfaces and provide examples of this framework for $\mathcal{C}_{obst}$ surfaces (UOBPRM) and medial axis surfaces (UMAPRM).

- We provide theoretical guarantees on the distribution of samples obtained by our method and prove that it preserves probabilistic completeness of sampling-based motion planners and performs stably with respect to changes in the narrow passage volume (UOBPRM) or in the surrounding obstacle volume (UMAPRM).

- We show applications of the uniform sampling framework in robotics (both UOBPRM and UMAPRM) and in biology (UOBPRM).

Portions of this research were previously published and presented. UOBPRM, which generates uniformly distributed configurations around $\mathcal{C}_{obst}$ surfaces, was published in the proceedings of the 2012 *IEEE International Conference on Intelligent*

Robots and Systems (IROS) [22]. UMAPRM, which uniformly samples the medial axis, was published in the proceedings of the 2014 *IEEE International Conference on Robotics and Automation* (ICRA) [73].

1.2 Outline

This dissertation is organized as follows. In Chapter 2, we provide background on sampling-based motion planning. We discuss several sampling methods that bias the sampling to specific surfaces (e.g., near $\mathcal{C}_{obst}$ boundaries or medial axis). In Chapter 3, we present a uniform sampling framework that provably distributes samples uniformly on surfaces in $\mathcal{C}_{space}$. We provide theoretical guarantees that it preserves probabilistic completeness of sampling-based motion planners and has a higher probability of sampling narrow passages. Chapter 4 presents UOBPRM as one specific instance from the uniform sampling framework that distributes samples uniformly around $\mathcal{C}_{obst}$ surfaces. We demonstrate that UOBPRM generates uniformly distributed samples and improves the efficiency to solve the motion planning problems. The relationship between UOBPRM and Gaussian PRM is compared both theoretically and experimentally. Chapter 5 presents another instance of the uniform sampling framework, UMAPRM, that generates the samples uniformly along the medial axis in $\mathcal{C}_{free}$. We evaluate the uniformity and the efficiency of UMAPRM against MAPRM. In Chapter 6, we show how we apply the uniform sampling framework to study the ligand binding affinity ranking problem. We present the results on three different target proteins and compare against experimentally determined ranking. We conclude with some final remarks in Chapter 7.

2. PRELIMINARIES AND RELATED WORK

In this chapter, we discuss motion planning preliminaries and existing sampling-based approaches. We limit the discussion to methods focused on planning on particular surfaces, such as narrow passages and along the medial axis of the free $\mathcal{C}_{space}$.

2.1 Motion Planning

A robot is a movable object that can be described by n parameters (*degrees of freedom*, DOFs) and each parameter represents an object component, such as position and orientation. All possible robot placements (or configurations) form an n-dimensional space, called configuration space ($\mathcal{C}_{space}$). Each robot configuration is represented as a point $\langle x_1, x_2, ..., x_n \rangle$ where x_i is the ith DOF in $\mathcal{C}_{space}$. All feasible robot configurations form $\mathcal{C}_{free}$, and $\mathcal{C}_{obst}$ is the union of all infeasible configurations. The motion planning problem is to find a path in $\mathcal{C}_{free}$ from a start configuration to a goal configuration. It is usually not feasible to compute the $\mathcal{C}_{obst}$ boundaries explicitly. However, we can utilize simple collision detection in the workspace (e.g., the actual space where the robot moves) to determine whether the configuration is valid or not.

Exact solutions are computationally infeasible, especially when the robot has many DOFs [5]. Some randomized algorithms have been developed to address this issue, e.g., sampling-based methods [40, 46] which solve many previously intractable problems. Specifically, Probabilistic RoadMap methods (PRMs) [40] construct a graph, or roadmap, to represent $\mathcal{C}_{free}$ by randomly sampling configurations and retaining valid ones. A simple local planner is applied to connect the configuration to its closest neighbors to form a roadmap. During the query process, the start and the goal configurations are added to the roadmap, connected by a local planner, and a

graph search algorithm, e.g, Dijkstra's algorithm, extracts the solution path. PRMs have been shown to map $\mathcal{C}_{free}$ efficiently but are not good at mapping some particular regions in $\mathcal{C}_{space}$, such as in narrow passages and along the medial axis [33].

2.2 Obstacle-Based Sampling

The probability of generating a sample in a particular region of $\mathcal{C}_{space}$ for the traditional uniform sampling [40] depends on the ratio of the region volume to the $\mathcal{C}_{space}$ volume. A narrow passage is a region in $\mathcal{C}_{space}$ of a volume so small that uniform random sampling is unlikely to generate any configuration in it [35] but that is important for planning, e.g., when solution paths are required to pass through it. Since $\mathcal{C}_{obst}$ surfaces define the boundaries of narrow passages, many PRM variants have been proposed to sample on those surfaces to increase coverage in these difficult regions. Here we discuss three approaches designed to address this issue: OBPRM, Gaussian PRM, and Bridge Test PRM.

2.2.1 OBPRM

Obstacle-Based PRM (OBPRM) [2] tries to generate samples close to obstacle surfaces. Algorithm 1 describes how OBPRM generates samples. It first finds a configuration c_{in} colliding with the obstacles. A random ray originated at c_{in} is selected and a free node c_1 is found along that ray. The boundary point is found by a bisection search between c_{in} and c_1. The boundary configurations will be kept as the roadmap nodes. Figure 1.3(d) shows samples generated by OBPRM.

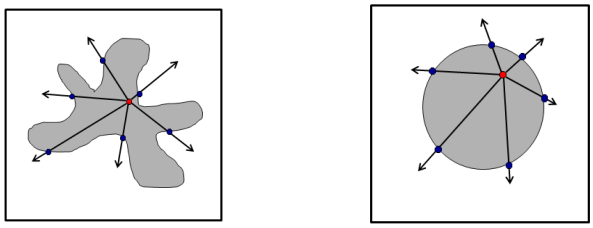
Although OBPRM can generate configurations close to $\mathcal{C}_{obst}$ surfaces, the node distribution is dependent on the $\mathcal{C}_{obst}$ shape and the position of the initial invalid configuration c_{in}, as shown in Figure 2.1. There is no perfect position for the initial colliding configuration to guarantee uniform node distribution if the shape of the $\mathcal{C}_{obst}$ is not spherical (see Figure 2.1(a)). Even if the $\mathcal{C}_{obst}$ is a sphere, the nodes cannot

Algorithm 1 OBPRM: Obstacle-Based PRM Sampler(n)

Input: A maximum number of attempts n and a step size t
Output: A set of nodes V near obstacles surfaces

```
1: V = ∅
2: for i = 0 → n do
3:    Randomly generate a point c_in in a C_obst
4:    Randomly select a point c_1 in C_space
5:    Let c_i = c_in
6:    while c_i not in C_free & c_i in C_space do
7:       Increment c_i by step size t along direction c_in c_1 (arrow)
8:    if c_i in C_space then
9:       Bisect between c_i and c_{i-1} to find free boundary point c_out
10:      Add c_out to V
11: return V
```

be uniformly distributed on the obstacle surfaces if the initial colliding configuration does not reside at the center of the $\mathcal{C}_{obst}$ (see Figure 2.1(b)). In particular, the portion of the surface closer to the initial colliding configuration will have a denser node distribution.



(a) (b)

Figure 2.1: The configuration distribution of OBPRM (in blue) is biased by (a) the shape of the $\mathcal{C}_{obst}$ and (b) the position of the initial colliding configuration c_{in} (in red).

Some work has been proposed to use workspace information to achieve a better node distribution [3]. The heuristics help to bias the initial colliding configuration

selection. The point representing an object (a robot or an obstacle) can be selected in different ways that will affect how the samples are biased. For example, using a random vertex to represent an object will bias node generation towards the portions of the object with more vertices. Selecting a triangle with probability proportional to its area and representing the object by a random vertex in that triangle can bias the sampling towards triangles with larger area. After selecting the points associated with the objects based on these heuristics, the robot is translated in order to coincide with the selection points of the robot and the obstacle and is rotated until finding the initial colliding configuration. Although the results show that these proposed heuristics can improve the node distribution, there are still no guarantees about the configuration distribution around $\mathcal{C}_{obst}$ surfaces.

2.2.2 Gaussian PRM

Gaussian PRM [13] attempts to generate configurations that are a Gaussian distance d away from the obstacle surfaces. A first configuration is randomly generated and the second is generated a Gaussian distance d away from the first configuration, where d is a user-specified parameter. If the validities of the two configurations are different, then the valid one will be retained as a node in the roadmap. Otherwise, both are discarded. Algorithm 2 illustrates the process and Figure 1.3(e) shows an example of Gaussian samples.

Gaussian PRM can be much slower and generate fewer samples than PRM in the same number of attempts since Gaussian PRM will discard many configurations that PRM would retain. Also, Gaussian PRM can be costly since it may be difficult to generate nodes with different validities. Roadmap quality is highly dependent on how d is selected. If d is too small, it is very likely that the configuration is too close to the $\mathcal{C}_{obst}$ causing collision. When d is large, the configurations are too far from

Algorithm 2 Gaussian PRM Sampler(n, d)

Input: A maximum number of attempts n and a distance d
Output: A set of nodes V a Gaussian distance d away from obstacle surfaces

```
1: V = ∅
2: for i = 1 → n do
3:    Randomly select a configuration c1
4:    Generate configuration c2 a Gaussian distance d along a random ray from c1
5:    if the validity of c1 and c2 are different then
6:       Add the valid one to V
7: return V
```

the $\mathcal{C}_{obst}$ surfaces. Gaussian PRM has an unknown node distribution.

2.2.3 Bridge Test PRM

Bridge Test PRM [32] has a similar node generation process to Gaussian PRM [13]. It also utilizes validity checking to bias sampling to the difficult regions, such as near $\mathcal{C}_{obst}$ surfaces and narrow passages. Algorithm 3 provides the pseudocode for Bridge Test PRM. It first generates an invalid configuration, and a second configuration is sampled a distance d away. If the second configuration is also invalid, the midpoint is found and its validity checked. The midpoint will be retained as a roadmap node if it is valid. Figure 1.3(f) shows an example roadmap generated by Bridge Test PRM.

Bridge Test PRM takes longer than OBPRM and Gaussian PRM since it needs to generate three consecutive samples in which the midpoint is valid and the endpoints are invalid. Bridge Test PRM also suffers from tuning the parameter d which can greatly affect the performance and the quality of the sampler. Finally, it has an unknown node distribution around $\mathcal{C}_{obst}$ surfaces.

2.3 Medial Axis Sampling

Most methods aim to simply find a feasible path. This may lead to paths with low clearance that may be a high risk for some applications. For example, OBPRM,

Algorithm 3 Bridge Test PRM Sampler(n, d)

Input: A maximum number of attempts n and a distance d
Output: A set of nodes V near obstacle surfaces

```
1: V = ∅
2: for i = 1 → n do
3:    Randomly select a configuration c_1
4:    if c_1 is invalid then
5:       Generate configuration c_2 a Gaussian distance d away along a random ray
         from c_1
6:       if c_2 is invalid then
7:          if the midpoint between c_1 and c_2 is valid then
8:             Add the midpoint to V
9: return V
```

Gaussian PRM, and Bridge Test PRM aim to focus the node density close to obstacles which increases the probability of sampling in the narrow passages but also results in samples that can be very close to the obstacles, making the extracted paths have a high risk of collision in the presence of localization errors. To compute high clearance paths, the Medial Axis Probabilistic Roadmap (MAPRM) [68, 49] generates configurations along the medial axis of $\mathcal{C}_{free}$. The medial axis is a set of points that are equidistant to two or more obstacles and are guaranteed to have maximal clearance. The medial axis is a *strong deformation retraction* which defines a one-to-one mapping between every point in $\mathcal{C}_{space}$ and the corresponding point on the medial axis and is thus a useful construction for motion planning.

Algorithm 4 outlines how MAPRM works. A random point q is first generated in $\mathcal{C}_{space}$. Depending on its validity, the configuration will be pushed either toward (if initially invalid) or away from (if initially valid) the closest point (witness point) on the $\mathcal{C}_{obst}$ boundaries until this closest point changes. A point on the medial axis has at least two witness points on $\mathcal{C}_{obst}$ boundaries while a point not on the medial axis has exactly one witness point. Therefore, the medial axis is crossed when there

is a change in the witness point. After detecting the medial axis, a binary search is applied to find the configuration residing on the medial axis, at a resolution ϵ. An example of MAPRM samples is shown in Figure 1.3(g). Note that it is not feasible to find the exact witness point in high dimensional space. In this case, approximate clearance and penetration computations are used [68].

Algorithm 4 Medial Axis PRM Sampler(n, t, ϵ)

Input: A maximum attempts n, a step size t, and a tolerance ϵ
Output: A set of nodes V along the medial axis

```
1: V = ∅
2: for i = 1 → n do
3:    Randomly generate a configuration q
4:    Find the witness point w of q on the obstacle boundaries
5:    if q is valid then
6:       Push q away from w at a step size t until the witness point changes
7:       Binary search finds the configuration c_ma with maximal clearance at resolu-
         tion ε
8:       Add c_ma to V
9:    else
10:      Push q toward w at a step size t until the witness point changes
11:      Binary search finds the configuration c_ma with maximal clearance at resolu-
         tion ε
12:      Add c_ma to V
13: return V
```

It has been shown that MAPRM can improve the sampling in narrow passages. The probability to sample inside the narrow passage with MAPRM depends not only on the volume of the free space in the narrow passage, but also on the volume of the surrounding obstacles since it pushes configurations regardless of their validity to the medial axis. However, it is still computationally expensive due to the clearance calculation, even with approximation.

The workspace medial axis can be used to bias sampling in the narrow passage [30, 72]. Both exact and approximate medial axis calculations can be applied to improve the sampling density in the narrow passage. However, they do not maximize the clearance in $\mathcal{C}_{space}$. Medial axis sampling is computationally intensive since it relies heavily on some expensive geometric computation.

A fast medial axis approximation is proposed in [50] which transforms the idea of finding the workspace medial axis to calculating the classification boundary for the labeling problem if each obstacle is labeled differently. It utilizes the max-margin optimization technique to push the configuration to the classification boundary and shows that it can generate samples on the medial axis more efficiently than MAPRM. However, none of these methods have any guarantee as to the resulting node distribution along the medial axis.

3. UNIFORM SAMPLING FRAMEWORK

Instead of sampling points in $\mathcal{C}_{space}$ and then filtering them or manipulating them (which is the trend of most sampling methods), our uniform sampling framework samples fixed length line segments and then identifies where (if any) the line segment crosses the target $\mathcal{C}_{space}$ surface. It is much easier to identify surface membership by detecting if the surface has been crossed than by evaluating membership at a single point in $\mathcal{C}_{space}$. Places where the line segment crosses the surface are retained in the roadmap as nodes. Which points are retained vary depending on the surfaces being sampled (e.g., $\mathcal{C}_{obst}$ or medial axis).

Section 3.1 provides the details of this framework. We theoretically prove that the uniform sampling framework provides guarantees of a uniform node distribution (Section 3.2) and that the framework is probabilistically complete (Section 3.3). We theoretically analyze its ability to generate configurations inside the narrow passage in Section 3.4 and find that it has a higher probability of sampling in the narrow passage.

3.1 Uniformly Generate Configurations in $\mathcal{C}_{space}$

We develop a methodology that uniformly samples specific surfaces in $\mathcal{C}_{space}$, e.g., near obstacle or along the medial axis, as long as surface membership can be determined by detecting the intersections between the line segment and the surfaces. A set of uniformly distributed fixed length line segments is generated by first sampling a random configuration c, and then selecting a direction at random $\overrightarrow{d}$, and extending the segment in direction $\overrightarrow{d}$ of length l from c. Then, roadmap nodes are identified as a result of some checking along the line segment.

Algorithm 5 shows this approach. As long as there is a criterion to find the

roadmap nodes by continuous checking the intersections between the line segment and the surfaces (line 5), this framework can generate uniformly distributed samples on any surface type in $\mathcal{C}_{space}$. `Intersect` is the function that finds the intersections between the line segment and the target surfaces. Depending on where the target surfaces are, `Intersect` has different checking criteria. Every valid crossing configuration along the line segment is stored as a roadmap node.

Algorithm 5 Uniform Sampling in Specific Surfaces of $\mathcal{C}_{space}$

Input: A maximum attempts n, a line segment length l, a step size t, and target surfaces R

Output: A set of uniformly distributed configurations V in R

1: Refine the bounding box
2: $V = \{\emptyset\}$
3: **while** $|V| < n$ **do**
4: Generate a uniformly distributed line segment s with fixed length l in $\mathcal{C}_{space}$
5: $V \leftarrow$ `Intersect`(s, l, t, R)
6: return V

3.1.1 Bounding Box Adjustment

The motion planning problem is solved within a bounding box in the environment. The target surface is not l-away from the bounding box, then segments that would yield potential samples may be disqualified. Hence, in order to maintain uniformity, we temporarily adjust the bounding box to ensure that the sampler has enough room to generate line segments with length l (line 1 in Algorithm 5) that could cover the full original environment. Since we still check the configuration validity with respect to the original bounding box, the original problem is not changed.

3.2 Uniformity

Here we prove that the configurations generated by the uniform sampling framework (Algorithm 5) are uniformly distributed on any specific target surface of $\mathcal{C}_{space}$. Since the line segments are generated uniformly in $\mathcal{C}_{space}$, the samples which are found on these line segments are also uniformly distributed.

Theorem 1. *Given a $\mathcal{C}_{space}$, the probability of finding an intersection point p from a line segment of length l chosen uniformly at random and some specific surface R in the $\mathcal{C}_{space}$ is constant throughout $\mathcal{C}_{space}$.*

Proof. Let $\mathcal{C}'_{space}$ be the adjusted space where the line segments are sampled. As shown in Figure 3.1, p is a point on R, S is the sphere centered at p with radius l, and $(c, l\overrightarrow{d})$ is a line segment with length l where c is a random configuration and $\overrightarrow{d}$ is a random direction. $R_{p,\epsilon}$ is the portion of R that is contained in a ball of radius ϵ centered at p, i.e., $R_{p,\epsilon} = R \cap B_{p,\epsilon}$, where $\epsilon > 0$. p is on $(c, l\overrightarrow{d})$ if and only if S contains c and $\overrightarrow{d}$ intersects $R_{p,\epsilon}$. Therefore, the probability P_R that the line segment intersects $R_{p,\epsilon}$ is equivalent to the probability that c resides in S and $\overrightarrow{d}$ intersects $R_{p,\epsilon}$, i.e., $P_R = P((c \in S) \wedge (\overrightarrow{d} \cap R_{p,\epsilon}))$. Given the conditions that c and $\overrightarrow{d}$ are both selected uniformly at random, P_R is uniform in the specific region R. □

Corollary 1. *For n randomly generated line segments of fixed length l, the probability of finding intersection points with some surface R is constant throughout $\mathcal{C}_{space}$. Since the probability of occurrence is the same, the distribution of the intersection points is uniform on R.*

3.3 Probabilistic Completeness

Here we prove that the planner given by the uniform sampling framework (Algorithm 5) is probabilistically complete.

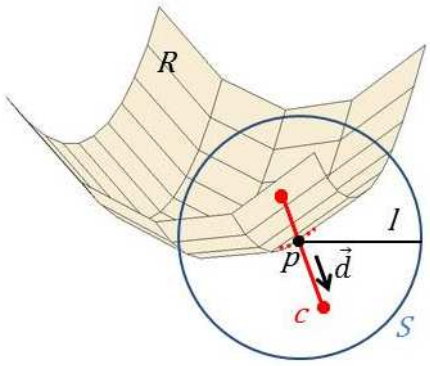


Figure 3.1: R is the target surface where uniform samples are distributed. The red dashed line represents the small portion $R_{p,\epsilon}$ on R where the line segment $(c, l\,\overrightarrow{d})$ crosses R. p is the intersection between the line segment and R. Therefore, the probability of a line segment intersecting the target surface is the probability that one endpoint of the line segment resides in a sphere with radius l centered at p and the $\overrightarrow{d}$ intersects $R_{p,\epsilon}$.

Theorem 2. *Let $a, b \in \mathcal{C}_{free}$ such that there exists a path γ between a and b lying in $\mathcal{C}_{free}$. Then, the probability that the planner correctly answers the query (a, b) after generating n configurations is given by*

$$Pr[(a,b)Success] = 1 - Pr[(a,b)Failure] \geq 1 - \left\lceil \frac{2L}{t} \right\rceil e^{-\sigma t^d n},$$

where L is the length of the path, t is the step size of the planner. $B_1(\cdot)$ is the unit ball in $\mathbb{R}^d$ and $\sigma = \frac{\mu(B_1(\cdot))}{2^d \mu(\mathcal{C}_{free})}$ where μ denotes the volume of a region of space.

Proof. Let $m = \left\lceil \frac{2L}{t} \right\rceil$ so that there are m points on the path $a = x_1, ..., x_m = b$ such that $dist(x_i, x_{i+1}) < t/2$. Let $y_i \in B_{t/2}(x_i)$ and $y_{i+1} \in B_{t/2}(x_{i+1})$. Then, the line segment $\overline{y_i y_{i+1}}$ must lie inside $\mathcal{C}_{free}$ since both end points lie in the ball $B_t(x_i)$. Let $V \subset \mathcal{C}_{free}$ be a set of n configurations generated uniformly distributed by the planner. If there is a subset of configurations $\{y_i, ..., y_m\} \subset V$ such that $y_i \in B_{t/2}(x_i)$, then a path from a to b will be contained in the roadmap. Let $I_1, ..., I_m$ be a set of

indicator variables such that each I_i witnesses the event that there is a $y \in V$ and $y \in B_{t/2}(x_i)$. It follows that the planner succeeds in answering the query (a, b) if $I_i = 1$ for all $1 \leq i \leq m$. Therefore,

$$Pr[(a,b)Failure] \leq Pr\left(\bigvee_{i=1}^{m} I_i = 0\right) \leq \sum_{i=1}^{m} Pr[I_i = 0]$$

The events $I_i = 0$ are independent since the samples are independent. The probability of a given $I_i = 0$ is computed by observing that the probability of a single randomly generated point falling in $B_{t/2}(x_i)$ is $\frac{\mu(B_{t/2}(x_i))}{\mu(\mathcal{C}_{free})}$. It follows that the probability that none of the n uniform, independent samples falls in $B_{t/2}(x_i)$ satisfies

$$Pr[I_i = 0] = \left(1 - \frac{\mu(B_{t/2}(x_i))}{\mu(\mathcal{C}_{free})}\right)^n.$$

Since the sampling is uniform and independent, then

$$Pr[(a,b)Failure] \leq m \times \left(1 - \frac{\mu(B_{t/2}(\cdot))}{\mu(\mathcal{C}_{free})}\right)^n.$$

However,

$$\frac{\mu(B_{t/2}(\cdot))}{\mu(\mathcal{C}_{free})} = \frac{(\frac{t}{2})^d \mu(B_1(\cdot))}{\mu(\mathcal{C}_{free})} = \sigma t^d,$$

where $\sigma = \frac{\mu(B_1(\cdot))}{2^d \mu(\mathcal{C}_{free})}$. We know that $(1-\beta)^n \leq e^{-\beta n}$ for $0 \leq \beta \leq 1$. Therefore,

$$Pr[(a,b)Failure] \leq m \times \left(1 - \frac{\mu(B_{t/2}(\cdot))}{\mu(\mathcal{C}_{free})}\right)^n \leq m \times e^{-\frac{\mu(B_{t/2}(\cdot))}{\mu(\mathcal{C}_{free})} n} = m \times e^{-\sigma t^d n} = \left\lceil \frac{2L}{t} \right\rceil e^{-\sigma t^d n}$$

□

3.4 Uniform Sampling in Passages

In this section, we examine the effectiveness of the uniform sampling framework to generate samples in $\mathcal{C}_{space}$ passages. The uniform sampling framework generates uniformly distributed configurations on the target surfaces by computing the intersections between a fixed length line segment and the target surface as shown in Algorithm 5. Below, we show that the probability for the uniform sampling framework to generate samples in a passage is dependent only on the surface area of the target surface in that passage and is independent of the volume of the passage.

We use the following notation:

- $SA(R)$ represents the surface area of the region R, where $R \in \mathcal{C}_{space}$.
- C_{RN} is the portion of the target surface C_R in the passage C_N.

Lemma 1. *The probability for the uniform sampling framework to generate configurations in a passage is correlated with the surface area of the target surface in the passage, $SA(C_{RN})$.*

Proof. As discussed in Section 3.2, the uniform sampling framework is proved to generate configurations that are uniformly distributed on the target surface. Therefore, the probability that the uniform sampling framework generates configurations in C_N is

$$P_{Uniform} = \frac{SA(C_{RN})}{SA(C_R)} \tag{3.1}$$

That is, the probability for the uniform sampling framework to sample in a passage is related to the proportion of the surface area of C_R that lies in the passage. □

Corollary 2. *The probability of generating samples in a passage does not depend on the volume of the passage.*

Proof. By Lemma 1, if the surface area of the target surface in the passage remains the same, then the uniform sampling framework is expected to generate the same number of samples in the passage, regardless of the volume of the passage. □

4. UNIFORM OBSTACLE-BASED PRM (UOBPRM)*

UOBPRM [22] is one instance of the uniform sampling framework which generates uniformly distributed samples near $\mathcal{C}_{obst}$ surfaces by simply defining an appropriate checking between the fixed length line segments and $\mathcal{C}_{obst}$ surfaces.

This is discussed in detail in Section 4.1. Section 4.1.1 illustrates the approach we use to temporarily adjust the bounding box for maintaining the uniformity. A discussion about the relationship between Gaussian PRM and UOBPRM is given in Section 4.2. Section 4.3 evaluates the sample distribution and efficiency of UOBPRM against PRM, Gaussian PRM, Bridge Test PRM, and OBPRM. Additionally, we experimentally show that UOBPRM has better performance in sampling in narrow passages compared to PRM, and Gaussian PRM and UOBPRM perform similarly with particular parameter settings.

4.1 Detecting Surface Membership

UOBPRM samples uniformly distributed configurations around obstacle surfaces by identifying the intersections between line segments and $\mathcal{C}_{obst}$ boundaries. This is done by applying validity checks to all intermediate configurations on the line segment. All validity changes indicate surface intersections and result in roadmap nodes. Thus, there can be more than one intersection between the segment and the obstacles, as shown in Figure 4.1. This feature allows UOBPRM to generate multiple nodes per segment when possible making it more efficient than other obstacle-based methods.

*The description of the method and some experimental results are reprinted with permission from "UOBPRM: A uniformly distributed obstacle-based PRM" by H. Y. Yeh, S. Thomas, D. Eppstein, N. M. Amato, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2655-2662 [22] ©2012 IEEE.

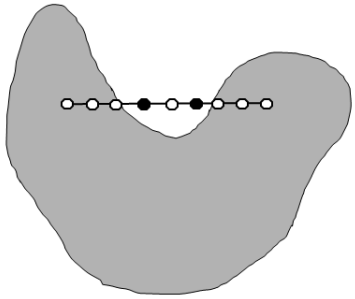


Figure 4.1: Finding intersections between the line segment and the obstacle by checking the validity of intermediate configurations along segment. The valid one is retained at every validity change. Here, the valid nodes that are retained are solid [22].

Algorithm 6 describes in detail how UOBPRM analyzes the line segments to find the intersections with the obstacle surfaces. The line segment length l and the step size t determine how close the configurations are to the obstacle boundaries and impact the efficiency of UOBPRM. The configurations are at most t-away from the $\mathcal{C}_{obst}$ surfaces. Therefore, the smaller the t is, the closer the configurations are to the $\mathcal{C}_{obst}$ surfaces. t needs to set based on the environment. It is usually the same as the resolution for collision detection in local planning. l and t play an important role in affecting the time for UOBPRM to generate nodes. If l is large and t is small, UOBPRM generally takes a long time since there are more intermediate configurations to check.

The specific surfaces R for UOBPRM in Corollary 1 where the uniform configurations are distributed is near $\mathcal{C}_{obst}$ as shown in Figure 4.2. The probability to find a line segment $(c, l\,\overrightarrow{d}\,)$ which crosses the target surface at point p is uniform throughout the environment since the fixed length line segment is distributed uniformly in the whole space.

Algorithm 6 UOBPRM `Intersect`(s, l, t, R)

Input: A line segment s of length l, a step size t, and target surfaces R
Output: A set of intersections I

```
1: R ← C_obst
2: for i = 1 → (l/t) do
3:     Generate node c_i along s
4:     if validity(c_i) ≠ validity(c_{i+1}) then
5:         Add the valid one to I
6: return I
```

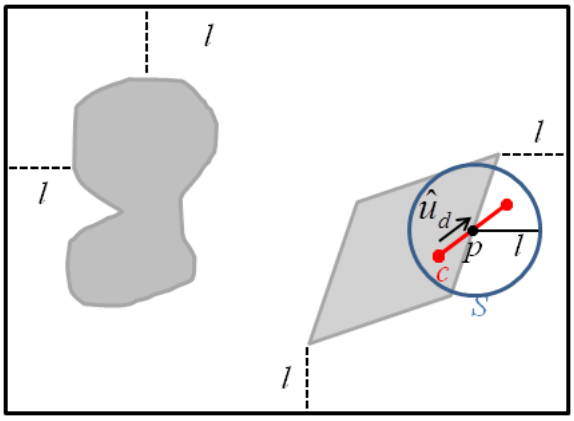


Figure 4.2: The target surface R is the $\mathcal{C}_{free}$ around $\mathcal{C}_{obst}$. p is the intersection between the line segment $(c, l\,\overrightarrow{d}\,)$ and the target surface which will be retained as a roadmap node. Since the line segments are uniformly distributed in the $\mathcal{C}_{space}$, the intersections found in R along the line segments are also uniformly distributed [22].

4.1.1 Bounding Box Adjustment

For UOBPRM, if the distance between the bounding box and the obstacles is less than l, then segments which would yield points on the $\mathcal{C}_{obst}$ surfaces may be disqualified. Figure 4.3 shows an example when the bounding box needs to be adjusted to maintain uniformity for UOBPRM.

UOBPRM adjusts the bounding box based on the information from the workspace

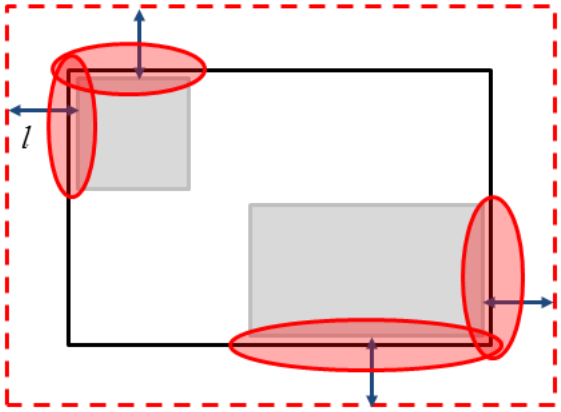


Figure 4.3: In this example, the uniformity guarantee is broken for UOBPRMbecause the original bounding box (solid line) is too close to the $\mathcal{C}_{obst}$ in the upper left and bottom right corners. This restricts the line segments that can be placed in the red regions. The bounding box is extended to the dashed line to allow all segments of length l that could intersect the $\mathcal{C}_{obst}$.

obstacles as described in Algorithm 7. The bounding box for each workspace obstacle is found first, and then a new bounding box in the workspace is determined that is the union of them all. This new bounding box expands each dimension by $l + r$ where l is the line segment length and r is the robot diameter, providing a bounding box which ensures that UOBPRM has enough space to generate line segments with length l around $\mathcal{C}_{obst}$ surfaces. We do not directly expand the original bounding box by $l + r$ since it may provide a bounding box larger than needed and may waste time on generating line segments without finding any intersection between the line segment and $\mathcal{C}_{obst}$. Figure 4.4 shows an example in which UOBPRM would suffer if the updated bounding box is directly expanded from the original one.

4.2 UOBPRM v.s. Gaussian PRM

Gaussian sampling [13] was proposed to improve the coverage of the difficult parts of $\mathcal{C}_{space}$ by only retaining samples near $\mathcal{C}_{obst}$ surfaces. A free configuration is added

Algorithm 7 Refine Bounding Box for UOBPRM Sampler

Input: The length l of the line segments used in sampling, maximum robot diameter r, and a set of obstacles O

Output: A new bounding box denoted by $min'\{x, y, z\}$ and $max'\{x, y, z\}$

1: original bounding box = $\{min\{x, y, z\}, max\{x, y, z\}\}$
2: $l = l + r$
3: $min_O\{x, y, z\} = min(min_o\{x, y, z\}; \forall o \in O)$
4: $max_O\{x, y, z\} = max(max_o\{x, y, z\}; \forall o \in O)$
5: $min'\{x, y, z\} = max(min_O\{x, y, z\} - l, min\{x, y, z\})$
6: $max'\{x, y, z\} = min(max_O\{x, y, z\} + l, max\{x, y, z\})$

to the roadmap only if there is an invalid configuration nearby. Gaussian sampling is like a filter for uniform sampling that reduces the samples in $\mathcal{C}_{free}$. Gaussian sampling employs the same uniform sampling. However, it discards some valid samples that PRM will keep in order to ensure the roadmap nodes are close to $\mathcal{C}_{obst}$ surfaces.

In terms of the method presented here, Gaussian sampling can be thought of as generating line segments of length d where d follows the Gaussian distribution (μ, σ) and, when the validity of the endpoints differs, identifying the valid endpoint as a roadmap node. UOBPRM instead generates line segments with fixed length l and finds the roadmap nodes when there is a validity change between two neighboring configurations along the line segment with a step size t. Gaussian sampling can behave like UOBPRM with some parameters setting: when they generate line segments with the same length and UOBPRM only checks the validities of the two endpoints. Thus, UOBPRM and Gaussian sampling are identical if $l = \mu = t$ and $\sigma \to 0$. When the length of the line segment (d in Gaussian sampling and l in UOBPRM) is small, these two methods are expected to behave very similarly. The running time for applying these methods is determined by the expected number of trials to collect n nodes in the roadmap. In the following lemma, we illustrate that UOBPRM with line segment length l and step size t has the same performance as Gaussian sampling

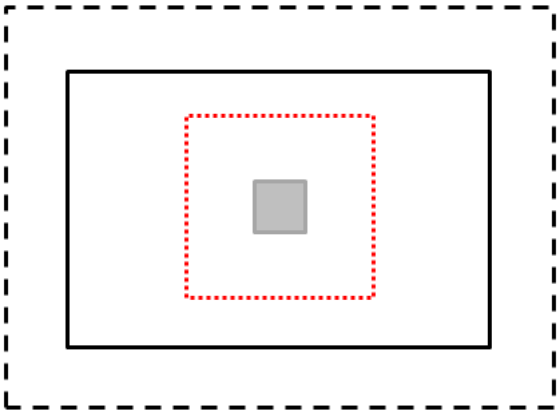


Figure 4.4: The example environment where directly expanding the bounding box by the line segment length l wastes time in generating line segments that can not find intersections in the targeted region. The solid line shows the original bounding box, the dashed line shows the bounding box directly expanded by l and the dotted line is the adjusted bounding box we perform.

with Gaussian distribution (μ, σ) when $l = \mu$, t converges to l and σ converges to 0.

Lemma 2. *The expected number of trials to obtain one roadmap node by UOBPRM with line segment length l and step size t and Gaussian sampling with Gaussian distribution (μ, σ) are the same if $l = \mu = t$ and $\sigma \to 0$.*

Proof. Since σ converges to 0, the length of every line segment generated by Gaussian sampling is very similar and is close to μ. Thus, both Gaussian sampling and UOBPRM generate line segments of the same length since $\mu = l$. When $t = l$, UOBPRM only checks the validity for the two endpoints along the line segment. In this case, both methods only check line segment endpoints of line segments of length $\mu = l$. The starting points of line segments for both methods are selected uniformly at random. Therefore, UOBPRM and Gaussian sampling perform very similarly when $l = \mu = t$ and σ converges to 0. □

Corollary 3. *The expected number of trials to obtain n roadmap nodes by UOBPRM*

with line segment length l and step size t and Gaussian sampling with Gaussian distribution (μ, σ) are the same if $l = \mu = t$ and $\sigma \to 0$.

Note that the previous Corollary is an extreme case for UOBPRM. Generally, Gaussian PRM is slower than PRM because while PRM randomly generates a configuration and adds the configuration to the roadmap if it is valid, Gaussian PRM only adds one configuration between two configurations and discards both of them if their validities are the same. However, UOBPRM may be faster than Gaussian PRM since UOBPRM can potentially generate more than one configuration from one line segment but Gaussian PRM gets at most one configuration from a line segment.

4.3 Experiment Results

In this section, we show some experimental results regarding node distribution and some motion planning problems for UOBPRM. All sampling methods are implemented in the C++ motion planning library developed in the Parasol Lab at Texas A&M University which contains a number of PRM variants and uses a distributed graph data structure from the Standard Template Adaptive Parallel Library (STAPL) [61], a C++ library designed for parallel computing.

The results show that UOBPRM is able to generate uniformly distributed configurations in the targeted surfaces in $\mathcal{C}_{space}$ (e.g., around $\mathcal{C}_{obst}$ surfaces) while other methods cannot. The computational cost for UOBPRM is comparable to other methods, and even less in some cases. We demonstrate that UOBPRM provides more stable performance with respect to the narrow passage width as compared to other obstacle-based sampling methods and it has higher probability to sample inside the narrow passage. UOBPRM can solve the motion planning problems more efficiently than other non-uniform sampling methods.

The results for UOBPRM are presented as follows:

- Planners studied — Section 4.3.1
- Uniformity analysis demonstrates the uniform distribution guarantee — Section 4.3.2
- Cost to generate configurations — Section 4.3.3
- Narrow passage analysis — Section 4.3.4
- Application to actual motion planning problems — Section 4.3.5
- Relationship between UOBPRM and Gaussian PRM as Gaussian PRM is a special case of UOBPRM when the parameters are set as noted in Section 4.2 — Section 4.3.6

4.3.1 Planners Studied

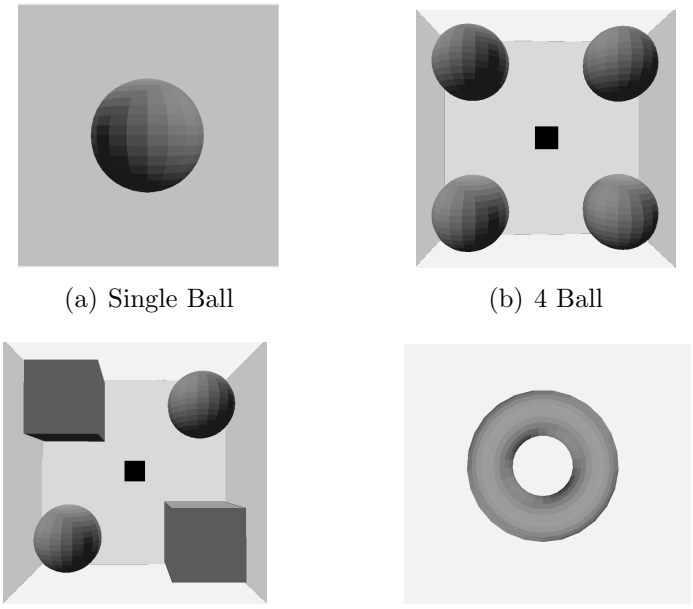
We compare five different sampling strategies: PRM [40], Gaussian PRM [13], Bridge Test PRM [32], OBPRM [2], and UOBPRM [22]. PRM is used as a control, and all the other sampling methods are developed for generating samples near obstacle surfaces.

The cost for node generation depends on how each sampling method generates configurations. Both Gaussian PRM and Bridge Test PRM are affected by a distance parameter d. OBPRM's cost is determined by the step size t and the cost of UOBPRM depends on l/t where l is the line segment length. We study different values for t, l, and d for these samplers. The results are averaged over 40 runs.

4.3.2 Uniformity

We study the performance of each sampler in different environments shown in Figure 4.5. Figure 4.5(a) has a unit radius ball obstacle at the center of an environment whose bounding box is $8 \times 8 \times 8$. Figure 4.5(b) has four unit balls placed on a

grid in a bounding box that is $8 \times 8 \times 8$. Figure 4.5(c) is a variant of Figure 4.5(b). It has a mixture of two unit balls and two cubes which are $2 \times 2 \times 2$. Figure 4.5(d) has a torus as the concave obstacle. The robot is a small cube for all environments.



(a) Single Ball (b) 4 Ball

(c) Mixed (d) Torus

Figure 4.5: Four environments are used to compare the distribution of samples produced by UOBPRM and other sampling methods. The robot is a small cube.

We first study the configuration distribution obtained by each method. In Figure 4.5(a), 4.5(b), and 4.5(c), we generate 4000 configurations with each sampling method and compute the node distribution by counting the number of configurations generated in each cell of a regular grid covering the environment by partitioning the space into 16 same sized cells. If the nodes are uniformly distributed, then the number of nodes should be proportional to the surface area for every region. In the torus environment (Figure 4.5(d)), we generate 4000 configurations and find the closest obstacle component each configuration belongs to. The obstacle is modeled as a

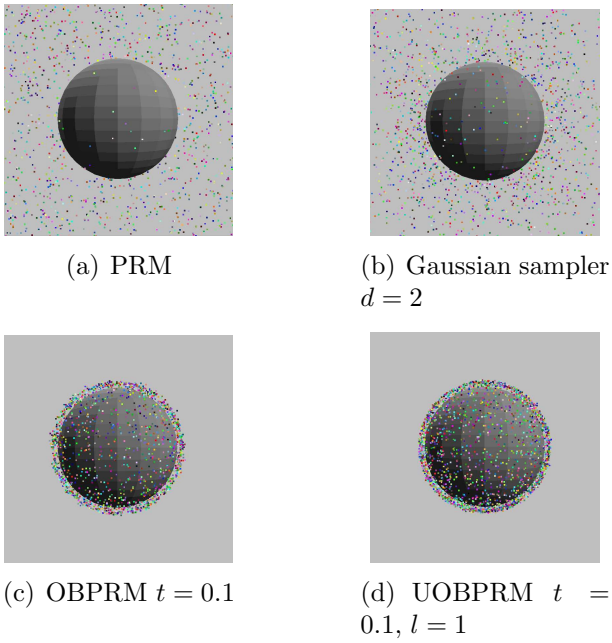
polyhedra composed of triangles. If the nodes are uniformly distributed, the number of configurations will be proportional to the area of the triangle.

4.3.2.1 A Single Ball Environment

The grid equally partitions the space into 16 cells. Starting from 1, the cells are indexed from left to right from top to bottom. Since the ball symmetrically occupies the center four cells (numbered 6, 7, 10, and 11), a similar number of configurations in these four cells is expected if the distribution is uniform around obstacle surfaces.

The configurations generated by PRM are dispersed throughout the environment, as shown in Figure 4.6(a). Figure 4.6(b) shows that Gaussian sampling still generates some configurations quite distant from the obstacle surfaces. Figure 4.6(c) shows the configurations generated by OBPRM with step size $t = 0.1$ and Figure 4.6(d) is for UOBPRM with step size $t = 0.1$ and line segment length $l = 1$. UOBPRM gives a more uniform distribution, and the configurations are closer to the obstacle surfaces compared to other sampling methods (see Figure 4.6).

Figure 4.7 compares the node distribution among different sampling methods. The red bars in Figure 4.7 show the percentage of configurations in the regions occupied by the ball and the blue bars represent the free space. If the configurations are uniformly distributed around obstacle surfaces, each red bar should result in 25% and the blue bar is 0%. OBPRM and UOBPRM are able to generate configurations near obstacles but PRM, Gaussian sampling, and Bridge Test sampling still have configurations scattered in the free space, especially PRM. Table 4.1 shows how the configuration distribution around ball and free space are for each sampling method. UOBPRM has the most uniformly distributed configurations among all sampling methods since its average is ideal and its standard deviation is the lowest. PRM, Gaussian PRM, and Bridge Test PRM still have some configurations distributed in



(a) PRM

(b) Gaussian sampler $d = 2$

(c) OBPRM $t = 0.1$

(d) UOBPRM $t = 0.1$, $l = 1$

Figure 4.6: Sample distribution example in the single ball environment. UOBPRM has the most uniformly distributed samples around the obstacle surfaces.

the free space.

Table 4.1: Average and standard deviation of ball and free space in single ball environment for different samplers. The ideal average for ball is 0.25 and free is 0. Error is calculated as the % difference to ideal.

Sampler	Ball			Free		
	Avg	Std	Error	Avg	Std	Error
PRM	0.0629	0.0079	-0.7484	0.0624	0.0071	n/a
Gaussian, $d = 0.2$	0.2328	0.0086	-0.0688	0.0057	0.0093	n/a
Bridge Test, $d = 3$	0.0411	0.0090	-0.8356	0.0696	0.0090	n/a
OBPRM	0.2500	0.0093	0.0000	0.0000	0.0000	0.0000
UOBPRM, $l = 1$	0.2500	0.0089	0.0000	0.0000	0.0000	0.0000

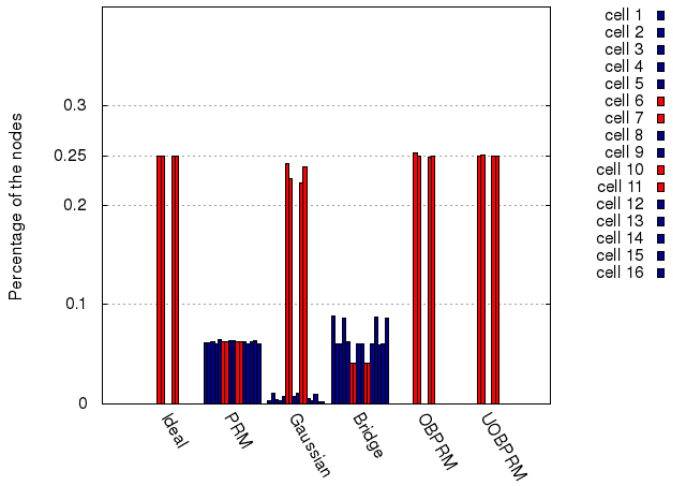


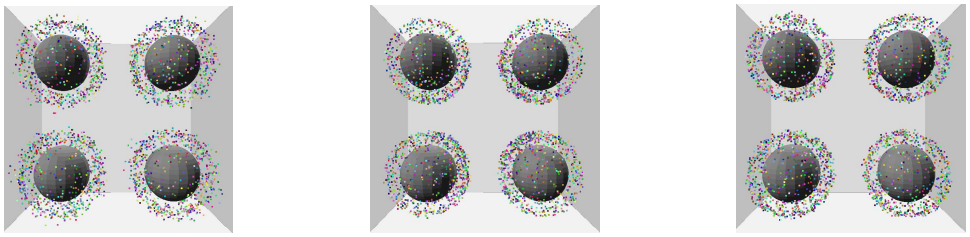
Figure 4.7: Distribution comparison of ball (red) and free (blue) regions in the single ball environment. Ideal percentage for ball is 25% and free is 0%.

4.3.2.2 Environment With 4 Balls of Equal Size

Here the step size t is 0.1 and the line segment length l is 1. Since we equally partition the space and the obstacle, we have a total of 16 cells with the same obstacle surface area. Ideally, a uniformly distributed obstacle-based sampler will generate same amount of the configurations in each cell.

Figure 4.8(a), 4.8(b), and 4.8(c) show the samples generated by Gaussian sampler, OBPRM and UOBPRM, respectively. Here UOBPRM and Gaussian sampling produce a distribution that is closer to uniform distribution than other sampling methods. As shown in Figure 4.8(b) and 4.8(c), OBPRM has fewer nodes on the boundary side than it should for a uniform distribution.

Figure 4.9 shows the node distribution comparison. Each color represents a different ball obstacle. If the distribution is uniform, each region will have 6.25% of the nodes. Table 4.2 shows how the configurations are distributed around each ball obstacle for various sampling methods. UOBPRM has the most uniformly distributed



(a) Gaussian sampler, $d = 0.2$

(b) OBPRM, $t = 0.1$

(c) UOBPRM, $t = 0.1$, $l = 1$

Figure 4.8: Sample distribution example in the 4 balls environment.

configurations comparing to other methods due to its lowest standard deviation.

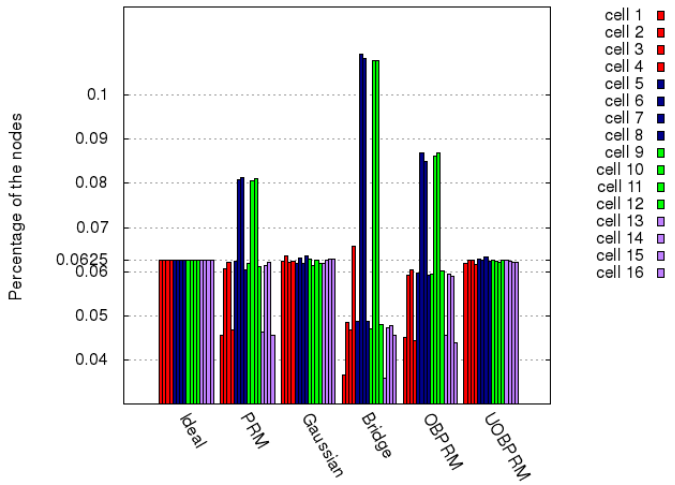


Figure 4.9: Distribution comparison in the environment with 4 balls of equal size, each ball is a different color. Ideal percentage is 6.25%. UOBPRM and Gaussian sampling generate more uniformly distributed samples than the others.

4.3.2.3 Environment With a Mixture of Balls and Cubes

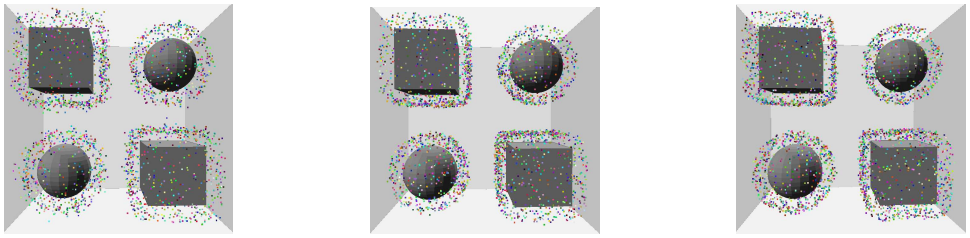
The step size t here is 0.025 and the line segment length l is 1. After generating 4000 nodes, we separate the environment into four regions where one region contains

Table 4.2: Average and standard deviation of each ball obstacle in the environment with 4 balls of equal size for different samplers. The ideal average for ball is 0.25. Error is calculated as the % difference to ideal.

Sampler	Ball 1			Ball 2			Ball 3			Ball 4		
	Avg	Std	Error	Avg	Std	Error	Avg	Std	Error	Avg	Std	Error
PRM	0.2151	0.0128	-0.1396	0.2848	0.0125	0.2500	0.2848	0.0126	0.2500	0.2153	0.0128	-0.1388
Gaussian, $d = 0.2$	0.2505	0.0029	0.0020	0.2503	0.0030	0.0012	0.2488	0.0032	-0.0048	0.2504	0.0034	0.0016
Bridge Test, $d = 3$	0.1975	0.0526	-0.2100	0.3152	0.0524	-0.2608	0.3107	0.0526	0.2428	0.1765	0.0523	-0.2940
OBPRM	0.2089	0.0153	-0.1644	0.2907	0.0150	0.1628	0.2925	0.0148	0.1700	0.2078	0.0156	-0.1688
UOBPRM, $l = 1$	0.2488	0.0038	-0.0048	0.2511	0.0034	0.0044	0.2495	0.0039	-0.0020	0.2494	0.0033	-0.0024

one obstacle, either a ball or a cube. The node distribution should be proportional to the obstacle surface area if the nodes are uniformly distributed around obstacle. The surface area for a ball is 4π and for a cube is 24. So ideally there should be about 1.9 times more nodes in the cube regions than in the ball regions. We again separate each obstacle into four same sized cells resulting in 16 cells for the entire environment.

Figure 4.10 shows that UOBPRM generates more uniformly distributed configurations within each cell than Gaussian sampling and OBPRM especially in the area close to the boundary.



(a) Gaussian sampler, $d = 0.2$

(b) OBPRM, $t = 0.025$

(c) UOBPRM, $t = 0.025$, $l = 1$

Figure 4.10: Sample distribution example in the mixture environment.

Figure 4.11 shows the node distribution for each sampler. The configuration percentage of the ball is colored in red and the cube is colored in blue. 4.31% is the ideal percentage of the nodes for the cells containing the balls and 8.19% is the ideal percentage of the nodes for the cells containing the cubes. The node distribution for UOBPRM is better than other sampling methods. Table 4.3 shows the configuration distribution around each obstacle (ball and cube respectively) for different sampling methods. UOBPRM has the most uniformly distributed configurations around obstacle surfaces since it has the lowest standard deviation.

Table 4.3: Average and standard deviation of ball and cube obstacle in the environment with a mixture of balls and cubes for different samplers. The ideal average for ball is 0.1718 and cube is 0.3282. Error is calculated as the % difference to ideal.

Sampler	Ball 1			Ball 2			Cube 1			Cube 2		
	Avg	Std	Error	Avg	Std	Error	Avg	Std	Error	Avg	Std	Error
PRM	0.2660	0.0136	0.5483	0.2670	0.0137	0.5541	0.2339	0.0131	-0.2873	0.2330	0.0134	-0.2901
Gaussian, $d = 0.2$	0.1948	0.0040	0.1339	0.1938	0.0042	0.1281	0.3057	0.0048	-0.0686	0.3057	0.0045	-0.0686
Bridge Test, $d = 3$	0.2407	0.0629	0.4010	0.2789	0.0632	0.6234	0.2408	0.0481	-0.2663	0.2396	0.0473	-0.2700
OBPRM	0.1897	0.0109	0.1042	0.1897	0.0109	0.1042	0.3107	0.0188	-0.0533	0.3099	0.0194	-0.0558
UOBPRM, $d = 1$	0.1734	0.0032	0.0093	0.1728	0.0029	0.0058	0.3266	0.0039	-0.0049	0.3271	0.0039	-0.0034

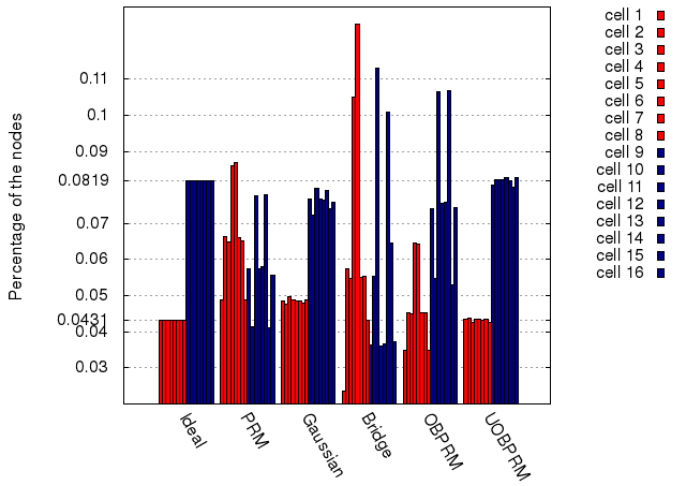


Figure 4.11: Distribution comparison of ball (red) and cube (blue) regions in the environment with a mixture of balls and cubes. Ideal percentage for ball is 4.31% and cube is 8.19%. The sample distribution for UOBPRM is the most uniform.

4.3.2.4 A Torus Environment

The outer radius of the torus is 2.5 and the inner radius is 1. The step size t is 0.01 and the line segment length l is 1 in this experiment. We generate 4000 nodes with each sampling method. For each configuration, we find the closest obstacle component it belongs to. The obstacle is modeled as a polyhedra composed by triangles. If the configurations are uniformly distributed around obstacle surfaces, the number of configurations should be proportional to the area of the triangles. This distribution error is calculated by the following where SA refers to the surface area:

$$\text{distribution error} = \left| \frac{SA(\text{triangle})}{SA(\text{model})} - \frac{\#\text{ of nodes belong to triangle}}{\text{total }\#\text{ of nodes}} \right|$$

The distribution error is used as a uniformity metric. The smaller value indicates a better uniform distribution.

Figure 4.12 shows the distribution error results normalized to PRM for each sampling method. UOBPRM can generate more uniformly distributed configurations around the obstacle surfaces compared to other sampling methods.

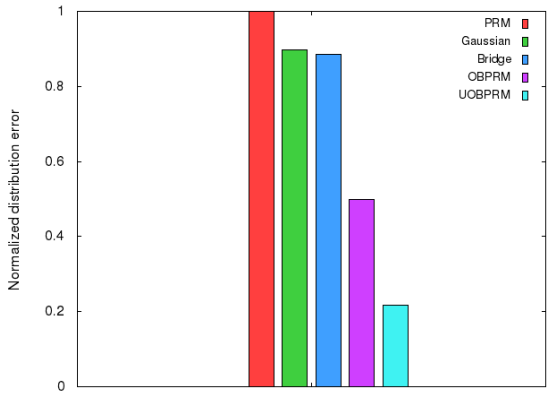


Figure 4.12: Normalized distribution error between different samplers. UOBPRM has the lowest distribution error among other samplers.

4.3.3 Cost

We are not only interested in the node distribution, but also in the cost of generating samples. PRM is fast when the $\mathcal{C}_{space}$ is free, but it does not work well in the difficult problems such as narrow passages. Gaussian sampling takes longer to generate samples because it must sample two configurations with different validities. Similarly, Bridge Test sampling needs to find a sequence of three samples such that the endpoints are invalid and the midpoint is valid. The cost for OBPRM is related to the step size t. The smaller the step size, the longer it takes to generate nodes. For UOBPRM, the time to generate nodes depends on both the length of the line

segment l and the step size t. If the segment is long and the step size is longer, then few intermediate configurations need to be tested. When the segment is short but the step size is small, then more intermediate configurations need to be checked. Therefore, the main factor determining the cost for UOBPRM is l/t.

Table 4.4: Generation time for various samplers and input parameters in the single ball environment [22].

Sampler	Parameter	Time (sec)
PRM	n/a	0.07
Gaussian	$d = 0.1$	22.91
	$d = 0.2$	7.08
Bridge Test	$d = 0.1$	128.58
	$d = 0.2$	86.73
OBPRM	$t = 0.025$	19.34
	$t = 0.05$	18.34
	$t = 0.1$	15.34
	$t = 0.2$	6.02
UOBPRM	$l/t = 2$	8.70
	$l/t = 4$	8.69
	$l/t = 5$	9.08
	$l/t = 10$	9.88
	$l/t = 20$	11.69
	$l/t = 40$	16.67

We examine the cost to generate configurations in two environments: the single ball environment (Figure 4.5(a)) and the Tunnel environment (Figure 4.16(a)). Tables 4.4 and 4.5 display the time for each sampling method to generate 1000 nodes in both environments. Both environments have similar trends. As expected, PRM takes the least amount of time. However, these samples are not distributed on the obstacle surfaces and PRM's performance degrades with increasing the problem dif-

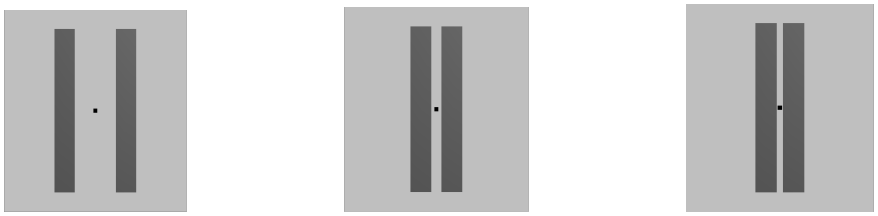
ficulty. Bridge Test sampling takes the longest time and is sensitive to d. It is because it is hard to sample two invalid points whose endpoint is valid. Depending on the parameters chosen, OBPRM and UOBPRM take the shortest amount of time. OBPRM increases generation time as the step size t is decreased. The time for UOBPRM is related to l/t. The generation time increases when l/t is increasing. Note that only UOBPRM makes any claim as to the distribution of samples on the obstacle surfaces, and it can do so with similar or even less computational time than the other methods.

Table 4.5: Generation time for various sampling methods and input parameters in the Tunnel environment [22].

Sampler	Parameter	Time (sec)
PRM	n/a	0.39
Gaussian	$d = 2$	28.66
	$d = 4$	44.39
	$d = 8$	82.41
Bridge Test	$d = 2$	85.86
	$d = 4$	172.29
OBPRM	$t = 0.025$	14.54
	$t = 0.05$	13.28
	$t = 0.1$	11.94
	$t = 0.2$	8.74
UOBPRM	$l/t = 2$	11.04
	$l/t = 4$	13.05
	$l/t = 5$	13.64
	$l/t = 8$	17.04
	$l/t = 10$	18.56
	$l/t = 20$	31.04
	$l/t = 40$	33.60
	$l/t = 80$	62.83

4.3.4 Narrow Passage Analysis

The probabilities to sample in the narrow passage for PRM, OBPRM, Gaussian PRM, and Bridge Test PRM are proportional to the volume of the narrow passage. However, the ability of UOBPRM to sample inside the narrow passage is less affected with respect to the changes in the narrow passage volume. In particular, the probability for UOBPRM to sample inside the narrow passage P_{UOBPRM} is $\frac{SA(\partial C_N)}{SA(\partial \mathcal{C}_{obst})}$, where $SA(\cdot)$ is the surface area of a region and C_N is the narrow passage in $\mathcal{C}_{space}$. Here we conduct two sets of experiments in the Passage 1, 2, and 3 environments (Figure 4.13). Passage 1 is an easy problem that contains a lot of free space. Passage 2 and Passage 3 have narrow passages where the robot can still rotate in Passage 2 but cannot in Passage 3. Note that the volume of the narrow passage varies while its surface area remains constant since the obstacles defining it remain unchanged.



(a) Passage 1 (b) Passage 2 (c) Passage 3

Figure 4.13: Environments that vary the narrow passage width. Passage 1 is the easiest problem and Passage 3 is the most difficult problem. The robot is a small cube.

4.3.4.1 Fixed Actual Number of Samples in the Map

We generate 1000 samples in the map using PRM, Gaussian PRM, Bridge Test PRM, OBPRM, and UOBPRM samplings. The number of configurations inside the narrow passage and the node generation time are collected for each sampling method.

Note that since OBPRM and UOBPRM can generate more than one sample in one attempt, the results are normalized to maps containing 1000 samples if needed.

Figure 4.14(a) first shows how many nodes are inside the narrow passage. When the passage width decreases, the probability of sampling inside the passage decreases for PRM, Gaussian PRM, and OBPRM. Bridge Test PRM always generates samples inside the narrow passage due to its nature. However, if the line segment cannot bridge the gap between obstacles (e.g., Passage 1 environment in Figure 4.13(a)), Bridge Test PRM is not able to generate any sample. The performance of UOBPRM is consistent with respect to the narrow passage width since the probability for UOBPRM to sample inside the narrow passage is affected by the surface area of the narrow passage.

Figure 4.14(b) shows the node generation time for each method sampling in three environments. PRM and Gaussian PRM are fast but their ability to sample inside the narrow passage highly depends on the volume of the narrow passage as shown in Figure 4.14(a). Bridge Test PRM again has the problem that its line segment length is problem dependent. It can only generate a sample when the line segment is long enough to bridge the obstacles. The node generation time for UOBPRM is relatively stable compared to OBPRM.

We also compute the percentage of samples that lie in the narrow passage for UOBPRM as shown in Figure 4.14(c). The surface area ratio between the narrow passage and the $\mathcal{C}_{obst}$ is 0.4444. Our result shows that 42.95% of the UOBPRM configurations are in the narrow passage which is very close to the ideal percentage of 44.44%.

4.3.4.2 Fixed Number of Samples inside Narrow Passage

The second experiment we have is to measure how many configurations are needed to generate a fixed number of samples inside the narrow passage by each sampling method. Here we generate 100 configurations in the narrow passage by PRM, Gaussian PRM, Bridge Test PRM, OBPRM, and UOBPRM. The results are normalized again to maps with 100 configurations in the narrow passages if the sampling method can generate more than one configuration by one attempt.

Figure 4.15(a) shows how many configurations are needed for each sampling method to obtain 100 samples inside the narrow passage. As the passage width decreases, PRM, Gaussian PRM, and OBPRM need to generate more samples in order to generate 100 configurations in the narrow passage. Although the performance of Bridge Test PRM is stable, it cannot generate any sample in the Passage 1 environment since the line segment is too short. UOBPRM performs very stably among three environments, and it also takes fewer nodes than other obstacle-based sampling methods (except Bridge Test PRM since all samples Bridge Test PRM generates will be inside the passage).

Figure 4.15(b) shows the time each sampling method takes to generate 100 samples in the narrow passage. The node generation time increases when the problem gets harder for all sampling methods except UOBPRM. The probability for PRM, Gaussian PRM, Bridge Test PRM, and OBPRM to generate configurations inside the narrow passage depends on the volume of the narrow passage. However, the performance of UOBPRM is affected by the surface area of the narrow passage. Thus, UOBPRM is more stable with respect to the changes in the narrow passage width.

We also compute the percentage of samples that lie in the narrow passage for UOBPRM. There should be about 44.44% of UOBPRM configurations in the narrow

passage by calculating the surface area ratio between the narrow passage and the $\mathcal{C}_{obst}$. Our experimental result shows that 43.00% of the UOBPRM samples lie in the narrow passage which is very close to the ideal percentage of 44.44%.

4.3.5 Motion Planning

In addition to the configuration distribution, we are also interested in how well the sampling method solves an actual motion planning problem. We study two difficult motion planning problems: a heterogeneous Tunnel environment as shown in Figure 4.16(a) which has various narrow passages with different widths and a Z-Tunnel environment (Figure 4.16(b)) which contains a narrow passage that a robot needs to traverse through. We try to use different sampling methods and the straightline local planner to find a path between the start and the goal configurations which reside in the free space at the two ends of the environment. The more uniform the configurations are, the faster the sampler is able to find a path by fewer nodes and edges.

Table 4.6 shows the average results in the heterogeneous environment (Figure 4.16(a)) for both a rigid body and a 4-link linkage robot by using different sampling methods with various parameters to solve the problem. UOBPRM with $t = 1$ performs the best since it needs the least configurations and edges to find the query path. PRM is not good at solving this kind of difficult problem. For UOBPRM, since a small step size t generates configurations closer to the obstacle surfaces, it needs more configurations and longer time to solve the problem. For the linkage robot, Bridge Test PRM takes more than one hour to find the solution path.

Table 4.7 shows the average results for solving the query in the Z-Tunnel environment by various sampling methods. UOBPRM outperforms other sampling methods by using fewer nodes and edges to efficiently solve the problem. Gaussian sampling,

Table 4.6: Time required to solve the heterogeneous Tunnel environment query by different robots for various sampling methods and input parameters. There are two types of robots: R for rigid body and L for linkage robot.

Robot	Sampler	# Nodes	# Edges	Time (sec)	CD Calls
R	PRM	7345.60	92259.00	1433.55	1017504.20
	Gaussian $d = 0.6$	1240.40	11381.60	39.68	147283.50
	Bridge Test $d = 4$	969.00	12031.50	11.13	125304.90
	OBPRM	763.00	5839.30	17.26	91578.10
	UOBPRM $l = 2, t = 0.2$	4790.10	51881.30	850.91	556786.00
	UOBPRM $l = 2, t = 0.25$	1507.20	13198.00	69.09	155927.80
	UOBPRM $l = 2, t = 0.4$	777.60	6237.70	22.13	63387.00
	UOBPRM $l = 2, t = 0.5$	908.10	7828.00	25.78	69271.20
	UOBPRM $l = 2, t = 1$	104.50	388.60	1.70	5769.30
L	PRM	566.90	2918.40	565.80	101407.30
	Gaussian $d = 0.6$	280.00	1795.10	731.73	147053.60
	Bridge Test $d = 4$	420.80	3119.80	8675.55	278293.10
	OBPRM	356.70	2226.20	477.21	70205.70
	UOBPRM $l = 2, t = 0.2$	322.20	1211.60	3012.32	785714.20
	UOBPRM $l = 2, t = 0.25$	319.70	1113.40	2915.46	538420.50
	UOBPRM $l = 2, t = 0.4$	224.90	868.30	73.63	41140.00
	UOBPRM $l = 2, t = 0.5$	87.20	214.50	8.41	9064.40
	UOBPRM $l = 2, t = 1$	72.30	252.00	4.25	5039.20

Bridge Test sampling and OBPRM are obstacle-based sampling methods. Therefore, they can find the solution path faster than PRM which is not good at solving the problem with difficult area such as narrow passages.

4.3.6 UOBPRM and Gaussian Sampling Performance Comparison

We compare Gaussian sampling and UOBPRM in a relatively free environment with a unit radius ball obstacle at the origin and a bounding box of $8 \times 8 \times 8$ (shown in Figure 4.17).

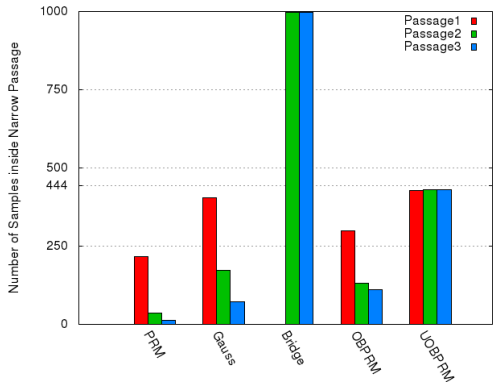
Figure 4.18 shows the node generation time normalized to Gaussian sampling for both methods to collect 4000 nodes with varying line segment lengths. When the line

Table 4.7: Time required to solve the Z-Tunnel environment query by different sampling methods.

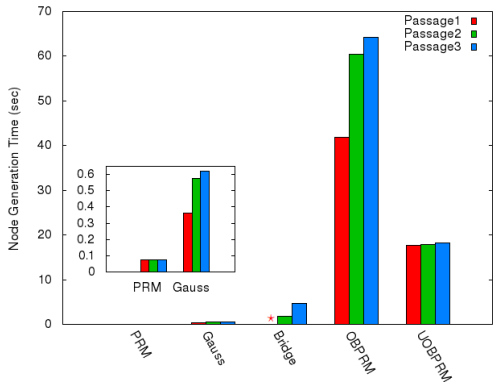
Sampler	# Nodes	# Edges	Time (sec)	CD Calls
PRM	10661.10	105701.40	1850.42	1331432.80
Gaussian $d = 0.2$	1894.00	15251.50	39.89	198697.10
Bridge Test $d = 3$	8359.20	73126.60	637.34	332410.20
OBPRM	3294.90	25720.00	503.51	267913.90
UOBPRM $l = 1, t = 0.01$	516.30	3936.60	29.59	117561.10

segment length is relatively small, both sampling methods have similar performance. As the line segment length increases, UOBPRM becomes increasingly more efficient than Gaussian sampling.

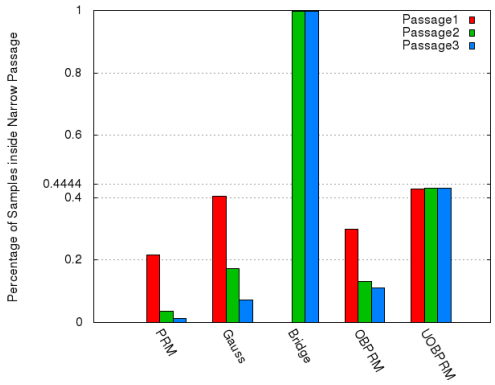
Figure 4.19 shows the configuration distribution for both UOBPRM and Gaussian sampling with various line segment lengths. The red bars in Figure 4.19 show the node distribution in the ball regions and the blue ones show the node distribution in the free regions. 25% is the ideal percentage in the ball regions if the configurations distributed uniformly around obstacle surfaces, and there shouldn't be any configurations in the free regions. When the line segment length is short, Gaussian sampling also generates uniformly distributed configurations around obstacle surfaces like UOBPRM (as shown in Figure 4.19(a) and Figure 4.19(b)). However, when the line segment is long, only UOBPRM achieves a uniform distribution (as shown in Figure 4.19(c)).



(a) Number of Samples

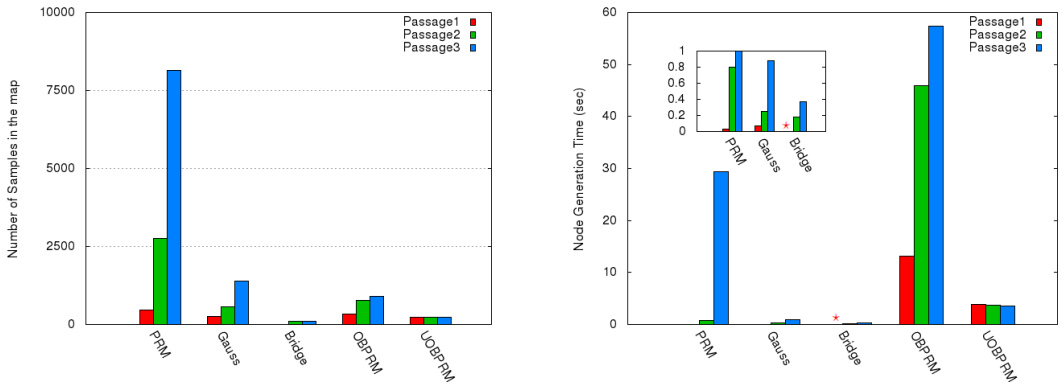


(b) Node Generation Time (sec)



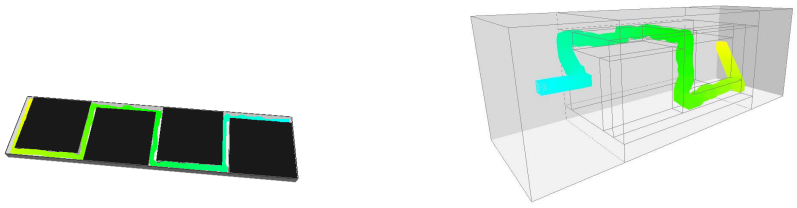
(c) Percentage of Samples

Figure 4.14: (a) Number of samples inside the narrow passage. (b) Time it takes to generate 1000 samples in the roadmap. $\star$ stands for infinity value as Bridge Test PRM is not able to generate any sample in the environment given the line segment length too short to bridge the gap between obstacles. (c) Percentage of samples in the narrow passage. The surface area ratio between the narrow passage and the $\mathcal{C}_{obst}$ is 0.4444. Only UOBPRM's performance is comparable.



(a) Number of Samples in the map (b) Node Generation Time (sec)

Figure 4.15: (a) Number of samples in the map in order to generate 100 samples inside the narrow passage. (b) Time it takes to generate the configurations. $\star$ indicates the infinity time that Bridge Test PRM needs in Passage 1 since the line segment length is not long enough to bridge the obstacles.



(a) Tunnel (b) Z-Tunnel

Figure 4.16: Two environments are used for the study of the motion planning problems. The robot is a small cube.

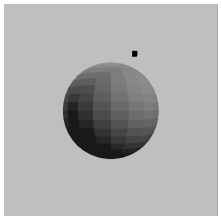


Figure 4.17: A relatively free environment is used to study the relationship between UOBPRM and Gaussian sampling. The robot is a small cube.

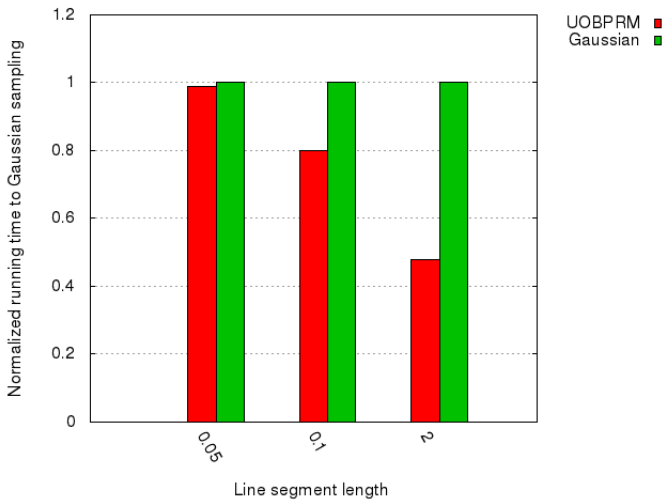
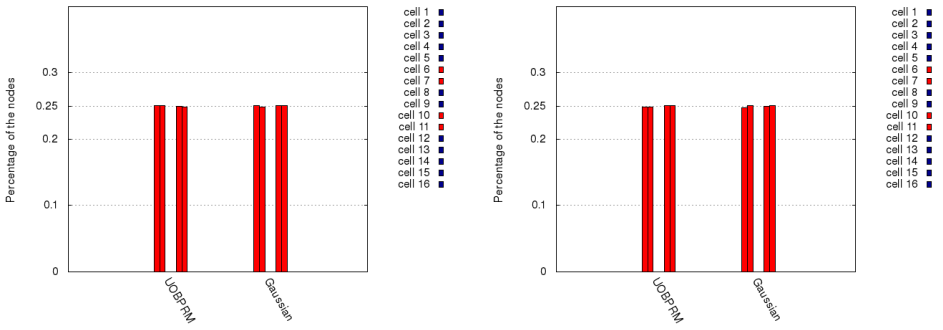
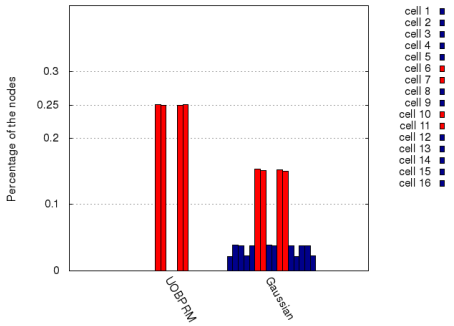


Figure 4.18: Time required to generate 4000 nodes by UOBPRM and Gaussian sampling with different line segment lengths (l in UOBPRM and d in Gaussian sampling). Step size t is equal to line segment length. Both methods perform similarly when the line segment length is short, and UOBPRM is more efficient than Gaussian sampling when the line segment length is long.



(a) $l = d = 0.05$ (b) $l = d = 0.1$



(c) $l = d = 2$

Figure 4.19: Distribution comparison of ball (red) and free (blue) regions. Ideal percentage of ball is 25% and free is 0%.

5. UNIFORM MEDIAL-AXIS PRM (UMAPRM)*

UMAPRM [73] is the instance of the uniform sampling framework that uniformly distributes samples along the medial axis by computing the crossing between the fixed length line segments and the medial axis surfaces.

We next describe in more detail how we generate samples uniformly distributed on the medial axis in Section 5.1. Section 5.2 demonstrates some experimental results including the node distribution, the performance to sample along the medial axis in the narrow passage, and the efficiency for solving some difficult motion planning problems.

5.1 Detecting Surface Membership

The medial axis is a set of points equidistant to two or more obstacles. Since every configuration on the sampled line segment has a corresponding closest obstacle, the medial axis is crossed when the closest obstacles changes. Figure 5.1 shows some configurations colored by the correspondingly closest obstacle.

Algorithm 8 outlines how UMAPRM detects the medial axis and uniformly samples on it. UMAPRM generates intermediate configurations at a step size t along the line segment and determine their closest obstacles. If there is a closest obstacle change between a consecutive pair of configurations, the medial axis has been crossed and a bisection search is used to find a configuration on the medial axis. Note that a single line segment might cross the medial axis multiple times and result in multiple medial axis points, as shown in Figure 5.2.

*The description of the method and some experimental results are reprinted with permission from "UMAPRM: Uniformly sampling the medial axis" by H. Y. Yeh, J. Denny, A. Lindsey, S. Thomas, N. M. Amato, 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 5798-5803 [73] ©2014 IEEE.

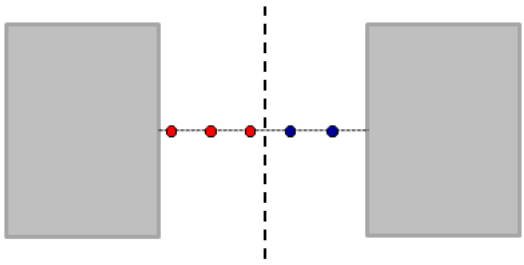


Figure 5.1: The configurations and their closest obstacles. The medial axis is given by the dashed line. Different colors represent different closest obstacles. The closest obstacle is changed when the medial axis is crossed.

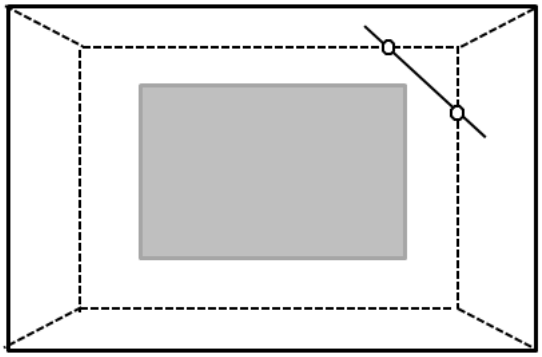


Figure 5.2: An example showing that more than one crossing point can be identified by a line segment.

The specific surfaces R for UMAPRM in Corollary 1 is along the medial axis of $\mathcal{C}_{free}$. Figure 5.3 shows an example from UMAPRM. Since the line segments are generated uniformly at random, the probability to find a line segment $(c, l\overrightarrow{d})$ which crosses the targeted surfaces at point p is uniform throughout the environment.

5.1.1 Bounding Box Adjustment

When the medial axis is too close to the bounding box, some line segments may not be considered because they are not fully in the bounding box, decreasing the

Algorithm 8 UMAPRM `Intersect`(s, l, t, R)

Input: A line segment s of length l, a step size t, and target surfaces R
Output: A set of intersections I

```
1: R ← medial axis in C_free
2: for i = 1 → (l/t) do
3:     Generate node c_i along s
4:     if closest obstacle(c_i) ≠ closest obstacle(c_{i+1}) then
5:         I ← BinarySearch(c_i, c_{i+1})
6: return I
```

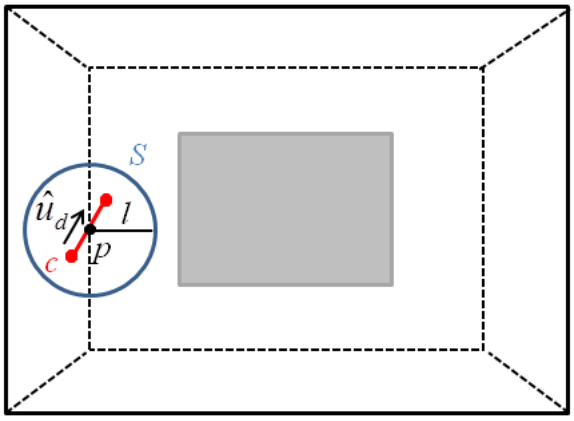


Figure 5.3: The target surface R is the $\mathcal{C}_{free}$ along the medial axis (dashed line). p is the intersection between the line segment $(c, l\overrightarrow{d})$ and the target surface which will be retained as a roadmap node. Since line segments are uniformly distributed in the $\mathcal{C}_{space}$, the intersections found in R along the line segments are also uniformly distributed [73].

probability for UMAPRM to generate line segments with length l. In particular, UMAPRM will invalidate the guarantee of uniformly sampling the medial axis when the medial axis is within distance l of the bounding box. To address this issue, UMAPRM temporarily adjusts the bounding box by expanding it by the line segment length $l+r$ to provide enough space for UMAPRM to generate uniformly distributed line segments in $\mathcal{C}_{space}$. Figure 5.4 shows an example when the bounding box needs

to be adjusted to maintain the uniformity for UMAPRM.

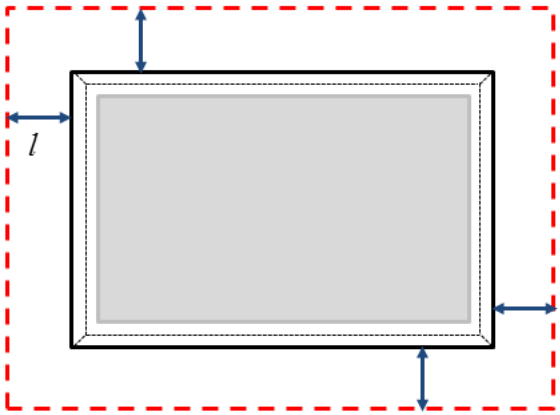


Figure 5.4: An example illustrating the situation when the uniformity guarantee is broken. The original bounding box (solid line) is too close to the medial axis, restricting the line segments that can be placed in the $\mathcal{C}_{space}$. The bounding box is extended to the dashed line to allow all segments of length l that could intersect the medial axis.

5.2 Experiment Results

In this section, we show some experimental results regarding node distribution and some motion planning problems for UMAPRM. All sampling methods are implemented in the C++ motion planning library developed in the Parasol Lab at Texas A&M University which contains a number of PRM variants and uses a distributed graph data structure from the Standard Template Adaptive Parallel Library (STAPL) [61], a C++ library designed for parallel computing.

The results show that UMAPRM is able to generate uniformly distributed configurations in the target surface in $\mathcal{C}_{space}$ (e.g., along the medial axis) while other methods cannot. This uniformity feature can benefit a real motion planning problem that other non-uniform sampling method is not able to solve. The computational

cost for UMAPRM is comparable to other methods, and even less in some cases. The probability for UMAPRM to sample in the narrow passage depends on the surface area of the medial axis in the narrow passage. Thus, UMAPRM is unaffected when there is a change in the surrounding obstacle volume while MAPRM's performance varies a lot due to the changes.

The results for UMAPRM are presented similarly as UOBPRM:

- Planners studied — Section 5.2.1
- Uniformity analysis demonstrates the uniform distribution guarantee — Section 5.2.3
- Cost to generate configurations — Section 5.2.4
- Narrow passage analysis — Section 5.2.5
- Application to actual motion planning problems — Section 5.2.6

5.2.1 Planners Studied

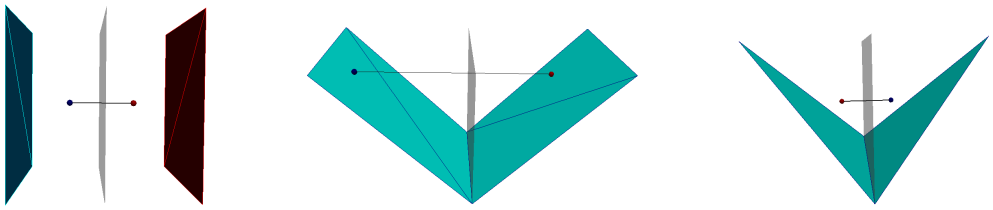
We compare UMAPRM, MAPRM, and PRM (uniform sampling) in this study. PRM is used as control, and the other two sampling methods are developed for generating samples along the medial axis. All methods use PQP [45] for collision detection and the Euclidean distance metric for distance calculation. We only consider point robots in this work. The results are averaged over 40 runs.

5.2.2 Implementation Detail for Point Robot

UMAPRM looks for crossings between the segments and the medial axis. Here we assume the robots are all point robots, so the medial axis of $\mathcal{C}_{space}$ is the same as of workspace. It first computes the closest witness point on the boundary of $\mathcal{C}_{free}$ and identifies which obstacle component it belongs to. Since the obstacles are modeled

as polyhedra composed of triangles, the obstacle component is either a vertex or a triangle. There are three cases causing c_i and c_{i+1} to be on the opposite sides of the medial axis as follows:

- Witness points belong to different obstacles (Figure 5.5(a)).
- Witness points belong to the same obstacle but are not on the adjacent obstacle components (Figure 5.5(b)).
- Witness points belong to the same obstacle and are on the adjacent obstacle components, but they are on the opposite side of a concavity in the model, e.g., two neighboring concave triangles (Figure 5.5(c)).



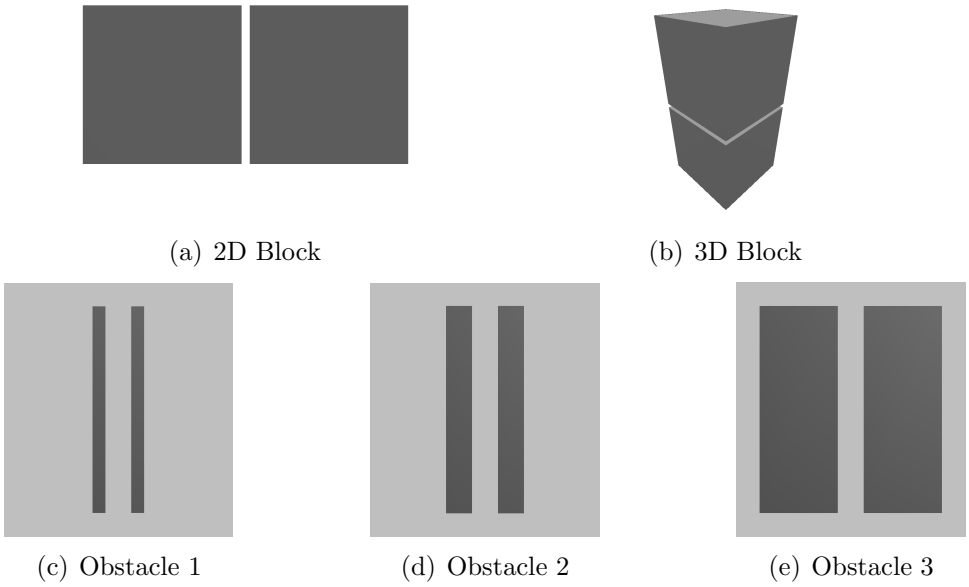
(a) (b) (c)

Figure 5.5: Three examples showing how UMAPRM finds configurations on the medial axis for a point robot by checking changes in closest triangles on obstacles. The grey face is the medial axis. The medial axis is crossed when (a) closest triangles are on different obstacles, (b) closest triangles are on the same obstacle but not adjacent to each other, or (c) neighboring concave triangles are on the same obstacle [73].

5.2.3 Uniformity

In this section, we provide a set of experiments showing the configuration distribution along the medial axis between UMAPRM and MAPRM. We show how certain environments (as shown in Figure 5.6(a) and Figure 5.6(b)) cause MAPRM

samples to be non-uniformly distributed, while UMAPRM samples do not have such bias.



(a) 2D Block (b) 3D Block

(c) Obstacle 1 (d) Obstacle 2 (e) Obstacle 3

Figure 5.6: (a, b) Two environments used to compare the distribution of UMAPRM and MAPRM. (c, d, e) Narrow passages of varying surrounding obstacle volume to compare sampling densities of UMAPRM and MAPRM. The robot we study in every environment is a point robot.

5.2.3.1 2D and 3D Environments With Two Unit Blocks

We first compare the configuration distribution generated by MAPRM and UMAPRM in simple 2D and 3D environments which contain a narrow passage created by two unit blocks (as shown in Figure 5.6(a) and 5.6(b)). We generate 1000 nodes by each sampling method along the segment of the medial axis between the blocks (we ignore the portion of the medial axis related to the boundary). Because the medial axis is simple (either a line or a plane), we compare with uniformly distributed points

generated by PRM along this structure. As a measure of uniformity, we compute the standard deviation of the distances between each node and its closest neighbor. If the nodes are uniformly distributed, this uniformity metric will be small.

Figure 5.7 shows that UMAPRM has the lowest average standard deviation in both environments, which implies that UMAPRM can generate more uniformly distributed nodes. Additionally, UMAPRM has roughly the same average as uniformly random distributed points along the medial axis plane.

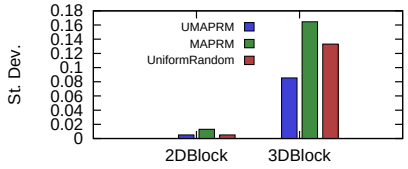
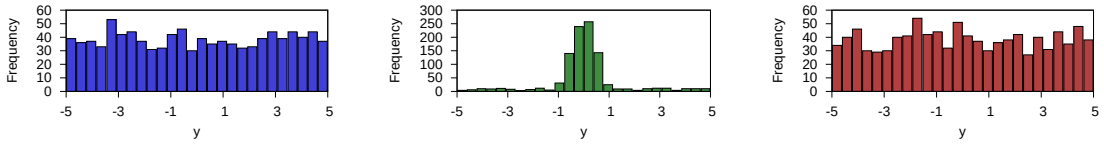


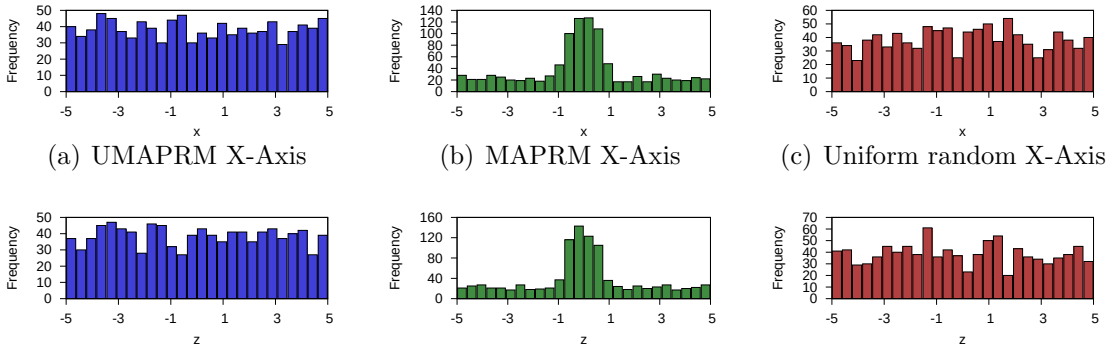
Figure 5.7: The average of standard deviations of distances between each node and its closest neighbor for roadmaps of 1000 samples between UMAPRM (green), MAPRM (blue), and uniform random sampling on the medial axis (red) [73].

Figure 5.8 and 5.9 show the sample distributions from the 1000 node roadmap generated by UMAPRM, MAPRM, and uniform random sampling on the medial axis. MAPRM is highly biased towards the area between the blocks in the environment, while UMAPRM is uniformly distributed along the medial axis for both 2D and 3D environments. MAPRM nodes are biased because MAPRM pushes samples away from the witness point on the $\mathcal{C}_{obst}$ boundary. Only samples whose closest points are on the corners of the block will be pushed towards the portion of the medial axis not covered by the block. The probability of this happening is much lower than for other portions of the medial axis. However, UMAPRM uniformly generates and analyzes line segments which do not have biases based on $\mathcal{C}_{obst}$ boundaries.



(a) UMAPRM (b) MAPRM (c) Uniform random

Figure 5.8: Distribution of 1000 samples generated by UMAPRM, MAPRM, and uniform random sampling in the 2D Block environment [73].



(a) UMAPRM X-Axis (b) MAPRM X-Axis (c) Uniform random X-Axis

(d) UMAPRM Z-Axis (e) MAPRM Z-Axis (f) Uniform random Z-Axix

Figure 5.9: Distribution of 1000 samples generated by UMAPRM, MAPRM, and uniform random sampling in the 3D Block environment [73].

5.2.4 Cost

We are also interested in how the costs of the sampling methods are affected when the surrounding obstacle width is varied. The three environments (Figure 5.6(c), Figure 5.6(d) and Figure 5.6(e)) have the same narrow passage width, while Obstacle 1 has the smallest obstacle volume and Obstacle 3 has the largest. We generate 1000 samples along the medial axis of the entire space and measure the node generation time for each method. The results are shown in Figure 5.10.

UMAPRM is also more consistent in the time it takes to generate samples in the narrow passages across the three environments, while MAPRM's efficiency is related to the distance for each node to be pushed to reach the medial axis. As obstacle

width increases, each node needs to traverse a smaller distance to the medial axis. Thus, MAPRM takes less time to generate successful samples in the Obstacle 3 environment than in the Obstacle 1 environment. However, UMAPRM is slightly affected in the third case. Although the change in the obstacle volume does not affect the probability of sampling in the narrow passage for UMAPRM, the total surface area of the medial axis has changed, which causes the average time for UMAPRM to generate samples to increase.

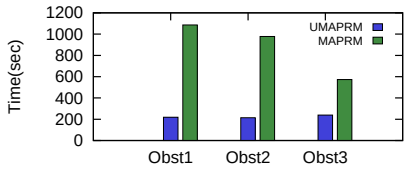


Figure 5.10: The time to generate 1000 samples for UMAPRM and MAPRM in Obstacle 1, 2, and 3 [73].

UMAPRM has a uniform distribution along the medial axis of the space while the distribution of samples in MAPRM can be highly non-uniform. The efficiency for UMAPRM is not affected by the volume of the surrounding obstacle volume. Conversely, MAPRM is hampered when the surrounding obstacle volume is small.

5.2.5 Narrow Passage Analysis

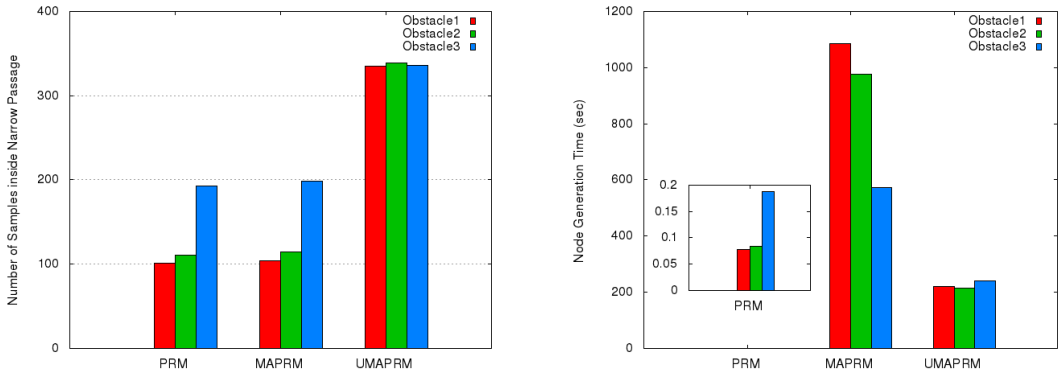
The probability to sample in the narrow passage for MAPRM is dependent on the volume of the narrow passage and the surrounding obstacle volume. However, the performance of UMAPRM sampling in the narrow passage is unaffected by the changes in the surrounding obstacle volume. Here we perform two sets of experiments for various environments in Figure 5.6(c), Figure 5.6(d) and Figure 5.6(e). Obstacle

1 has the smallest surrounding obstacle volume and Obstacle 3 has the largest. Note that the volume of the narrow passage is the same for these three environments.

5.2.5.1 Fixed Actual Number of Samples in the Map

We generate 1000 samples along the medial axis of the entire environment by PRM, MAPRM, and UMAPRM. The number of configurations inside the narrow passage and the node generation time are collected for each sampling method. Note that the results are normalized if needed since UMAPRM can generate more than one sample in a single attempt.

Figure 5.11(a) first shows how many nodes there are inside the narrow passage. When the surrounding obstacle volume increases, MAPRM is able to generate more samples in the narrow passage. UMAPRM consistently generates almost the same number of samples in the narrow passage regardless the changes since the surface area of the medial axis in the narrow passage is fixed.



(a) Number of Samples

(b) Node Generation Time (sec)

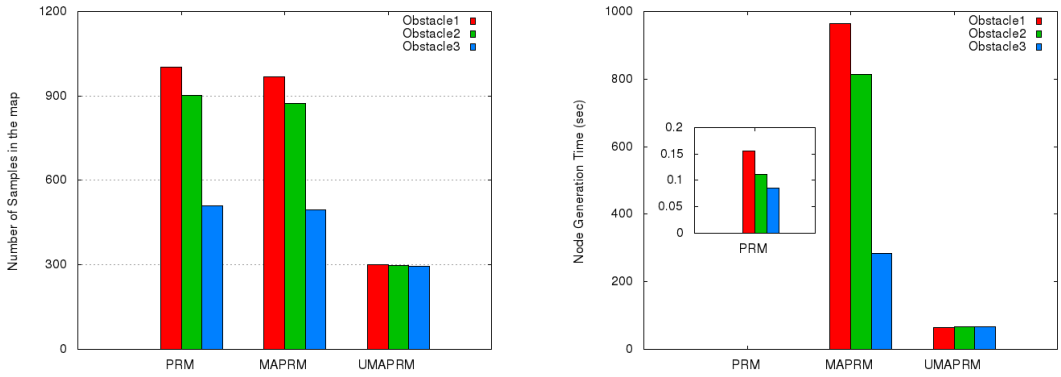
Figure 5.11: (a) Number of samples inside the narrow passage. (b) Time it takes to generate 1000 samples in the roadmap.

Figure 5.11(b) shows the node generation time for each method sampling in three

environments. MAPRM takes longer to generate samples than UMAPRM in the three environments. Also, the node generation time for UMAPRM is relatively stable compared to MAPRM.

5.2.5.2 Fixed Number of Samples inside Narrow Passage

The second experiment we have is to measure how many configurations are needed to generate a fixed number of samples inside the narrow passage for each sampling method. Here we generate 100 configurations in the narrow passage by PRM, MAPRM, and UMAPRM. The results are normalized again if the sampling method can generate more than one configuration in a single attempt.



(a) Number of Samples in the map (b) Node Generation Time (sec)

Figure 5.12: (a) Number of samples in the map in order to generate 100 samples inside the narrow passage. (b) Time it takes to generate the configurations.

Figure 5.12(a) shows how many configurations are needed for each sampling method in order to get 100 samples inside the narrow passage. As the surrounding obstacle volume increases, MAPRM needs fewer configurations in the map in order to have 100 configurations in the narrow passage. However, UMAPRM performs very stably among three environments and it always takes fewer nodes than

MAPRM.

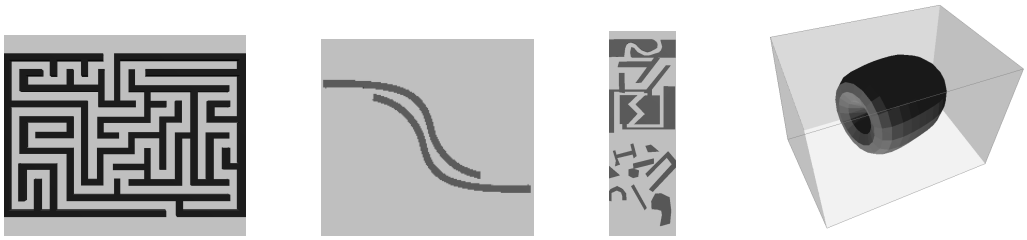
Figure 5.12(b) is the time each sampling method takes to generate 100 samples in the narrow passage. MAPRM spends less time to generate 100 configurations in the narrow passage when the surrounding obstacle volume increases since its performance depends not only on the narrow passage volume but also on the surrounding obstacle volume. On the other hand, the performance of UMAPRM is related to the surface area of the medial axis in the narrow passage. Thus, UMAPRM is more stable with respect to the changes in the surrounding obstacle volume.

5.2.6 Motion Planning

In this section, we compare how UMAPRM, MAPRM, and PRM can solve the planning problem by finding a path from a start to a goal configuration in the environment as shown in Figure 5.13. There is a long narrow passage in the 2DMaze environment (Figure 5.13(a)). In the STunnel environment (Figure 5.13(b)), the narrow passage is surrounded by thin obstacles. In the 2DHeterogeneous environment (Figure 5.13(c)), there are multiple narrow passages with different types. In the Bug Trap environment (Figure 5.13(d)), the robot needs to escape from the trap by traversing a small opening.

5.2.6.1 Time

We first study the efficiency of each sampling method to solve the query. The result is normalized to PRM. Figure 5.14 shows that UMAPRM takes less time to find the solution than MAPRM, but is slower than PRM in the 2DMaze and the 2DHeterogeneous environments. This is because PRM has fewer collision detection calls in these environments. In the STunnel environment, UMAPRM outperforms both MAPRM and PRM. PRM takes longer because the volume of the narrow passage is small compared to the rest of the planning space. MAPRM is also hampered



(a) (b) (c) (d)

Figure 5.13: Motion planning environments studied. The robot is a point robot for all environments. (a) 2DMaze. The start and the goal reside at the two ends in the free space. (b) STunnel. The start and the goal are in the top left and the bottom right corners. (c) 2DHeterogeneous. The start is in the top free space and the goal is placed in the bottom cluttered region. (d) Bug Trap. The objective is to get out of the trap through the narrow passage.

in this environment because the volume of the surrounding obstacle volume is still small compared to the rest of the planning space. In the Bug Trap environment, only UMAPRM is able to find the solution within the 10-hour running time limit.

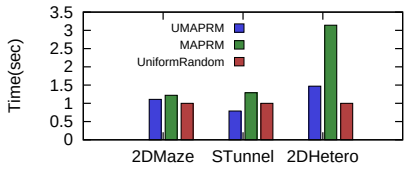


Figure 5.14: The time to solve the problem for UMAPRM, MAPRM, and PRM in different environments [73].

5.2.6.2 Clearance

In addition to the efficiency, we are also interested in the quality of the path by calculating the average path clearance for each sampling method. The average path clearance is the average of the edge clearances for each solution path. Figure 5.15 is the normalized results to PRM. It shows that UMAPRM can generate higher quality

paths than MAPRM and PRM in the 2DMaze and the STunnel environments. In the 2DHeterogeneous environment, the quality between UMAPRM and MAPRM is comparable. Since only UMAPRM is able to solve the query problem in the Bug Trap environment, there is no normalized quality result for this environment.

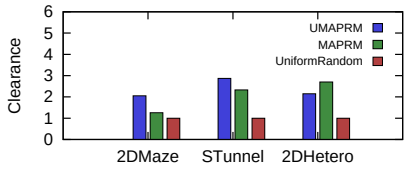


Figure 5.15: The average clearance of the path for UMAPRM, MAPRM, and PRM in different environments [73].

6. RANK LIGAND BINDING AFFINITY

In the pharmaceutical industry, the drug screening process rigorously tests potential drugs to select the most promising candidates for further study. Since the drug discovery process is quite costly [36], computationally screening drug candidates against target proteins is becoming increasingly important. Analysis of surfaces is essential to study the ligand binding problem. The uniform sampling framework presented in Chapter 3 is a strategy for sampling uniformly on a particular surface. Hence, the uniform sampling framework can be applied in drug screening.

The drug, or ligand, is a small molecule which attempts to bind to a specific site (called the binding site or the binding pocket) on the target protein, as shown in Figure 6.1. This interaction is essential to many biochemical processes as well as to the efficacy of various drug candidates. Drug design in the disease treatment is an inhibition case during binding. In AIDS (acquired immunodeficiency syndrome) treatment, the virus is no longer able to cause further infection only when the drug can successfully bind to the enzyme [29, 69]. The strength of this interaction is known as the ligand binding affinity. Ligands that bind with higher affinities have a higher likelihood to be more effective drugs that bind quicker, be more stable, and remain bound longer than ligands with low affinities. Thus, the pharmaceutical industry is always searching for ligands with high binding affinities to their particular protein target.

Many computational approaches have studied ligand binding. In particular, many robotics techniques that compute feasible motions for a robot have been applied to this domain by considering the ligand as the robot [57, 7, 23, 24]. However, none of these techniques rank ligands based on their affinity.

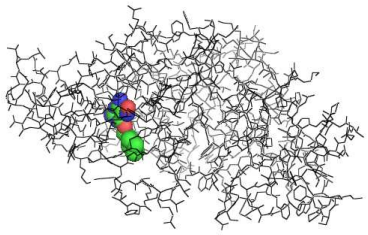


Figure 6.1: A protein (shown in wireframe) with a ligand (shown in spheres) bound inside it.

We present a method to rank binding affinity by sampling ligand conformations uniformly distributed over the target protein's surface, analyzing the resulting sample set, and computing an affinity based properties of these samples. Specifically, we use UOBPRM sampling [22] originally developed for robotic motion planning, to generate samples guaranteed to be uniformly distributed over the protein's surface, regardless of the complexity of that surface. We present two different affinity metrics and test their ability to correctly rank ligands as determined by experimental data on three different target proteins. We show that one of our metrics, in particular, can correctly rank all the ligands for all protein targets.

6.1 Preliminaries

In this section, we discuss some approaches that are used to compute ligand binding affinity, including experimentally and computationally (Section 6.1.1.1 and Section 6.1.1.2 respectively). Next, we explain how the motion planning framework can be adjusted to study molecule's motion in Section 6.1.2.

6.1.1 Ligand Binding Affinity

Ligand binding affinity quantifies the ability (or inability) of a particular ligand to bind, or dock, to a target protein. Figure 6.1 shows an example of a ligand (shown in spheres) bound to a target protein (shown in wireframe). The ligand binding pro-

cess, sometimes called molecular docking, can be described by two important models. The lock-and-key model assumes that there is a high degree of similarity between the shape of the protein and the ligand (Figure 6.2). The ligand with complementary geometry can trigger the binding process, like a key in a lock (Figure 6.2(a)). Otherwise, incompatible ligand fails to bind to the protein (Figure 6.2(b)). The lock-and-key model does not take the protein flexibility into account when discussing the binding process. The second model which considers that protein has some flexibility is called the induced-fit model. The ligand binding induces some protein's conformational change that results in a complementary fit between the protein and the ligand, see Figure 6.3.

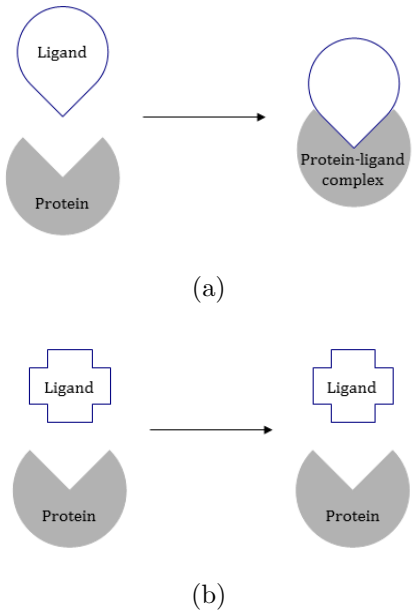


Figure 6.2: The lock-and-key ligand binding model: (a) A ligand successfully binds to the target protein due to complementary geometry and chemistry. (b) The ligand is incompatible and the protein-ligand complex cannot form.

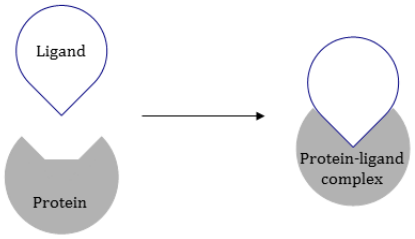


Figure 6.3: In the induced-fit model, the protein undergoes a conformational change when ligand binds to it. The shape of the ligand becomes complementary to the shape of the binding site after the ligand binds to the protein.

6.1.1.1 Experimental Approaches

Normally, binding affinity can be calculated by *IC50* and K_i where K_i can be derived by *IC50* [17]. *IC50* is the concentration required to achieve 50% inhibition of a biochemical function. It can be measured by linear regression which starts with some concentration of an agonist (a chemical compound that will activate some biological response when binding) and keeps increasing the concentrations of an antagonist to inhibit the response. The greatest value of the response will be normalized to 100% and all the other response values are computed as the percentage of the greatest response.

IC50 can also be computed by competition binding which is done by measuring the agonist with a radioactive isotope attached when increasing antagonist concentrations. *IC50* is the amount when there are 50% of the binding of the radio-agonist.

Unlike *IC50* changes depending on the experiment, K_i is an absolute value and is often referred to the inhibition constant of a drug. K_i is the concentration of the drug when 50% of the receptors is occupied and there is no radio-agonist present. K_i can be calculated by the following equation where $[L]$ is the concentration of the

radioligand used and K_d is the dissociation constant of the radioligand:

$$K_i = \frac{IC50}{(1 + \frac{[L]}{K_d})}$$

The lower the *IC50* or K_i values are, the more potent the drug is and the greater affinity it has for the receptor.

Many databases collect the protein-ligand interaction information. BindingDB [51] is a public database which provides ligands with different binding affinities for a specific target protein. Binding MOAD [8] and BioLiP [70] collect protein structures and their relevant ligands with experimentally determined binding data. An improved benchmark for molecular docking, DUD-E [54], includes ligands and their target proteins. More importantly, DUD-E provides decoys for each ligand which can be used to improve molecular docking screening.

6.1.1.2 Computational Approaches

Ligand binding experiments are accurate but expensive and labor-intensive [64]. In light of this, many computational molecular docking approaches have been developed which use Monte Carlo and Genetic Algorithms to predict protein-ligand binding. DOCK [44], AutoDock [53], Gold [28], and FTDock [39] are some such approaches. None of these approaches treats the ligand as fully flexible [65]. Moreover, the exploration space is very large and computation times are extremely long. In addition, the success ratio is often relatively low (2–20%) [66].

There are approaches which use motion planning to study protein-ligand binding [57, 7]. In [57], they first uniformly generate ligand conformations over the space and then sample more densely in the known binding site. Samples are connected together if there exist energetically feasible transitions between them. A low-potential

path is extracted from the resulting graph. This path represents a possible ligand binding path. In [7], obstacle-based sampling (OBPRM) [2, 3] and human input via a haptic device provide better quality ligand samples. They were better able to predict the ligand binding site on the protein surface. However, they did not specifically study ligand binding affinity.

Other approaches also focus on finding ligand access and exit pathways. Rapidly exploring random trees (RRTs) [47] is used in [23] to study how (R, S)-enantiomers exit the active site of *Burkholderia cepacia lipase*. An extended ML-RRT method is applied to compute the exit pathways of TDG from *lactose permease* (LacY) and the exit pathways of *carazolol* from β_2-*adrenergic receptor* in [24]. MoMA-LigPath [26] is a web server to simulate ligand unbinding process by ML-RRT method. Steered Molecular Dynamics [37], Random Acceleration Molecular Dynamics [52], and Monte Carol techniques [14] are used to study the ligand binding simulation. They also did not specifically rank ligands according to their binding affinity.

6.1.2 Modeling Molecular Motions

Similar to robots, molecules also have motions. Moreover, these motion are usually essential to activate many important mechanisms or even to cause some diseases. For example, a protein can fold to its final, stable three dimensional structure. When a protein misfolds into a different structure, it can possibly lead to many devastating diseases, such as Alzheimer's, Mad Cow, and Parkinson's disease [20]. When insulin binds to the insulin receptor, it will activate the formation of the insulin receptor substrate. This substrate will later cause some intracellular insulin effects, such as fat metabolism and glucose uptake [56].

As modeling motions has been well studied in robotics in the past several decades, we can surprising study many biology problems by changing the definition of the

robot and a valid robot placement with the same motion planning framework in traditional robotics [57, 4, 1, 62].

The overall strategy follows the general approaches presented in robotics, only with some modifications in order to adapt molecules to the framework as discussed in Section 6.1.2.1 and Section 6.1.2.2.

6.1.2.1 Molecule Model

The molecule is modeled as an articulated linkage robot. Depending on the motion that the protein can have, the degree of freedoms (DOFs) are assigned to the bond angles with some range (such as a revolute joint ranging between $[0, 2\pi)$).

6.1.2.2 Molecule Validity

A valid molecule configuration depends on the definition of a "collision-free" sample. The criteria can be a energetically stable structure or a molecule that has no geometric collision with another molecule (for example, a ligand and a protein cannot collide to each other).

6.2 Method

To rank ligand binding affinity, we first generate a set of ligand conformations uniformly distributed over the target protein surface using UOBPRM for each ligand under consideration. We then analyze the resulting ligand-protein conformation sample sets and compute various binding affinity metrics. We use the metrics to then rank the relative affinities of several different ligands to the same target protein.

We first discuss how we model the protein and the ligand in Section 6.2.1. We then present the methodology for using UOBPRM to rank binding affinity in Section 6.2.2. Finally, we present binding affinity metrics in Section 6.2.3

6.2.1 Protein and Ligand Models

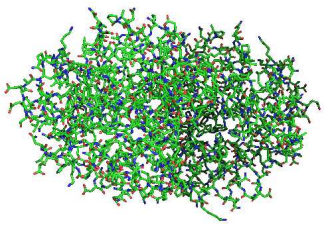
We model the protein as a rigid obstacle (as shown in Figure 6.4(c), Figure 6.5(c), and Figure 6.6(c)) and the ligand as a flexible linkage robot. Note that the protein "obstacle" (Figure 6.4(c) for 3W6H, Figure 6.5(c) for 4RRW, and Figure 6.6(c) for 4K5Y) is very similar to the protein viewed in spheres by PyMOL [25] (Figure 6.4(b) for 3W6H, Figure 6.5(b) for 4RRW, and Figure 6.6(b) for 4K5Y).

The ligand is modeled as a free base articulated linkage robot where the torsional movement of the bond is modeled by one DOF revolute joint ranging from 0 to 2π. A DOF value is assigned to a bond between two atoms only if the rotation between these two atoms can change the shape of the ligand. Therefore, we treat ring structures as rigid. Note that unlike most other work, the ligand is treated as completely flexible (apart from ring structures) instead of modeling the ligand as a static structure or only minimally flexible.

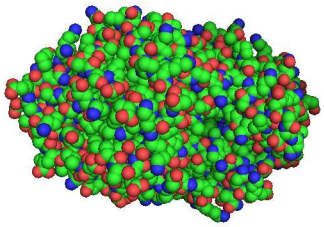
6.2.2 Using UOBPRM to Rank Binding Affinity

Since we assume the protein remains mostly static, we treat it as an rigid obstacle. We can then apply UOBPRM where the protein is the obstacle and the ligand is sampled around it. This generates a set of ligand samples uniformly distributed over the protein's surface, including any binding pockets or cavities.

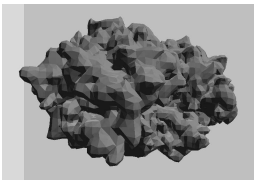
Note that there are several important parameters to set: the number of samples n, the length l of line segments, and the resolution t at which to check consecutive points along the line segments. (These are the same input parameters to the uniform sampling framework, see Algorithm 5). We discuss each in the following sections.



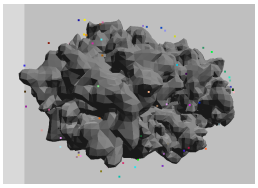
(a) Protein (in wireframe)



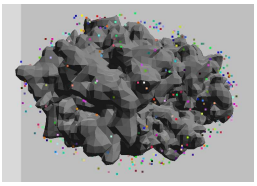
(b) Protein (in spheres)



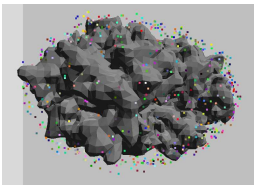
(c) Rigid Protein Obstacle



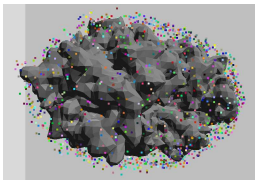
(d) 100 Ligand Samples



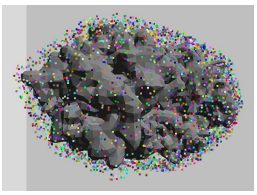
(e) 500 Ligand Samples



(f) 1000 Ligand Samples



(g) 2000 Ligand Samples



(h) 5000 Ligand Samples

Figure 6.4: Protein 3W6H in wireframe (a) and in spheres (b) viewed by PyMOL [25]. (c) The protein is modeled as a rigid obstacle. (d)-(h) Varying numbers of ligand samples (ligand centers of mass only shown).

6.2.2.1 The Number of Samples, n

We determine the appropriate number of UOBPRM ligand samples by checking whether the exposed residues are covered well enough based on a *coverage measurement*. The exposed residues are identified based on the relative solvent accessible area information. The DSSP (Define Secondary Structure of Proteins) file [38] is parsed to extract the ACC (accessibility) information. The relative solvent accessible area is computed by dividing ACC by the total surface area of the residue [21]. Those who had relative solvent accessible area below 20% are defined as buried while

others are exposed residues [19]. For every ligand sample, we identify which exposed residue it is close to and calculate the distance between each sample and its closest neighboring sample. The average of all the distances is computed. If the average distance is below certain percentage of the ligand diameter, we have enough samples to cover the protein surfaces. Otherwise, we need more samples to approximate the protein surfaces. Figures 6.4, 6.5, and 6.6 show the varying coverage with different values of n of the ligand over the protein surface for 3W6H, for 4RRW, and for 4K5Y, respectively. For clarity, only the ligand's center of mass is shown. Although it has been proven that the number of samples does not affect the uniformity [22], we need to have enough samples to cover the protein surface well in order to evaluate them and determine an affinity.

6.2.2.2 The Line Segment Length, l

The second input parameter to UOBPRM is the length of the line segments. However, in [22], it was proven that l does not affect the resulting distribution. Thus, we simply fix l to 10 in all the experiments here.

6.2.2.3 The Checking Resolution, t

The final parameter to consider is the step size t at which to check consecutive points along the line segment for surface membership. Because we are defining surface membership as near obstacle surfaces, t will greatly affect the distances ligand samples are to the protein surface. Here, we want the ligand samples as tight to the protein surface as possible to more accurately model the binding interaction. To do so, we set t to be the same resolution at which steric collisions are typically set.

6.2.3 Affinity Metrics

Given a set of UOBPRM ligand configurations, we would like to use them to approximate the ligand binding affinity with respect to the target protein. To compute such an affinity, we look at several different metrics. For each metric, we examine the minimum value, the average value of the minimum 1%, and the average value of the minimum 10%.

In this paper, we investigate the following affinity metrics:

- *Distance.* This calculates the distance between the center of mass of the target protein and the center of mass of each UOBPRM ligand sample. This is based off the idea that ligands with higher binding affinities are more likely to be buried deeper in the protein. Thus, the smaller the distance, the higher the affinity. Note that this metric requires no knowledge of the binding site location.

- *Energy.* The potential energy between the protein and the UOBPRM ligand sample measures the energetic compatibility of the complex. It is calculated as:

$$U = \sum_{\text{atom pair}i,j} A/r_{ij}^{12} - B/r_{ij}^{6} + E_{hydrophobic}$$

  where r_{ij} refers to the distance between protein atom i and ligand atom j. Parameters A and B are taken from [48]. Since the ligand searches for a stable low potential conformation during binding, the lower the energy, the higher the affinity. Note that this metric also requires no knowledge of the binding site location.

6.3 Experiment Results

We compare the ligand binding affinity ranking from our method to the experimentally determined affinity ranking order for three different target proteins: 3W6H with 3 ligands, 4RRW with 5 ligands, and 4K5Y with 4 ligands. Binding affinities were obtained from BindingDB [51], and proteins were obtained from the Protein Data Bank [9]. We apply UOBPRM on each protein-ligand pair to generate a set of ligand configurations around the target protein's surface and use the affinity metrics discussed in Section 6.2.3 to approximate the binding affinity.

6.3.1 Target Protein 3W6H

The first set of experiments contains 3 ligands with different binding affinities with respect to protein 3W6H (see Figure 6.4(a)). Figure 6.7 displays the ligands structure studied ordered by affinity ranking from best to worst. 3W6H is a human carbonic anhydrase I (hCAI) which is a lyase catalyzing the removal of some bonds from a compound [60]. Carbonic anhydrase converts carbon dioxide and water to bicarbonate. Thus, it helps balance the acid base in the body [58].

Table 6.1 shows the binding affinity ranking determined by the experiments and our affinity metrics. All ligands are identified by PubChem Database CID [12]. Here, experimental binding affinity is determined by the K_i value. We generate and analyze 5000 UOBPRM ligand samples.

The distance affinity metric is able to correctly capture the affinity ranking order as determined by experiment for all three statistics: minimum value, 1% average, and 10% average. The energy affinity metric performs well when only considering the average of the top 1% ligand samples. As more conformations are considered in the average, there is a greater likelihood that they will not all reside closely to the binding pocket.

Table 6.1: Comparison of published binding affinity ranking and approximated binding affinity ranking for 3W6H.

Ligand CID	Affinity (K_i nM)	Published Rank	Distance			Energy		
			Min.	1% Avg.	10% Avg.	Min.	1% Avg.	10% Avg.
768	5×10^{-4}	1	1	1	1	1	1	1
24530	12×10^{-4}	2	2	2	2	3	2	3
19366655	200×10^{-4}	3	3	3	3	2	3	2

6.3.2 Target Protein 4RRW

The second set of experiments has 5 ligands which bind to protein 4RRW with different strengths. The structures for the protein and 5 ligands are shown in Figure 6.8 ordered by affinity ranking from best to worst. Note that for this target protein, the ligand structures are much more complex than before (e.g., containing many ring structures and multiple branches). 4RRW is a cyclooxygenase-2 (COX-2) that helps synthesize prostaglandin in the body [10]. Cyclooxygenase (COX) is found when there is inflammation. Therefore, the inflammation and the pain can be relieved by inhibiting COX [27]. Here, we generate and analyze 7000 UOBPRM ligand samples.

Table 6.2 shows the binding affinity ranking determined by the experiments and our affinity metrics. All ligands are identified by PubChem Database CID [12]. Here, experimental binding affinity is determined by the *IC50* value.

Again, we see that the distance affinity metric performs well. It was able to capture the correct affinity ranking order most of the time, except for the top 10% average statistic. The ligands will be less likely placed in the binding pocket when we analyze more conformations, particularly if the binding pocket is relatively small. The energy affinity metric performs similarly as it did for the previous experiment

Table 6.2: Comparison of published binding affinity ranking and approximated binding affinity ranking for 4RRW.

Ligand CID	Affinity (*IC50* nM)	Published Rank	Distance			Energy		
			Min.	1% Avg.	10% Avg.	Min.	1% Avg.	10% Avg.
46224069	2.07	1	1	1	1	1	1	1
11441881	30	2	2	2	2	2	2	4
3672	1100	3	3	3	4	4	3	2
2244	13900	4	4	4	3	3	4	3
71460125	¿ 50000	5	5	5	5	5	5	5

that it can correctly rank the binding affinity order when considering the top 1% average statistic.

6.3.3 Target Protein 4K5Y

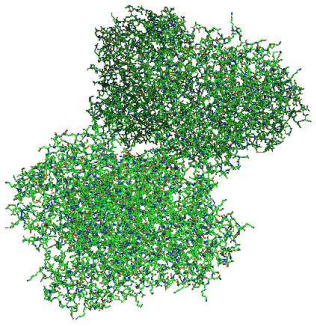
The third set of experiments has 4 ligands with different binding affinities with respect to protein 4K5Y (as shown in Figure 6.9 ordered by affinity ranking from best to worst). 4K5Y is Corticotropin-releasing hormone receptor 1 that is in the family of G protein-coupled receptor. This protein will activate stress-related hormone. Thus, it is important in the treatment of depression and anxiety disorder [31]. We generate and analyze 5000 UOBPRM ligand samples for this protein.

Table 6.3 shows the binding affinity ranking determined by experiments and our affinity metrics. All ligands are identified by PubChem Database CID [12].

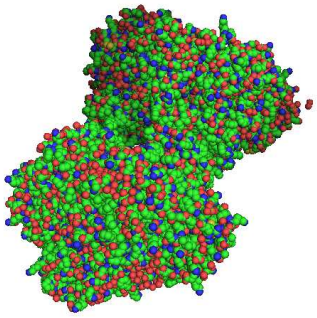
The distance affinity metric again performs well. It was able to correctly rank the affinity order most of the time, except for the top 10% average statistic. Similarly, the energy affinity metric does well in the top 1% average statistic.

Table 6.3: Comparison of published binding affinity ranking and approximated binding affinity ranking for 4K5Y.

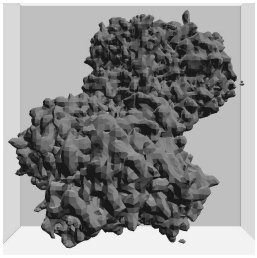
Ligand	Affinity	Published	Distance			Energy		
CID	(K_i nM)	Rank	Min.	1% Avg.	10% Avg.	Min.	1% Avg.	10% Avg.
10595854	10	1	1	1	1	1	1	2
11065415	526	2	2	2	3	3	2	1
10180472	2800	3	3	3	2	2	3	3
1087955	¿ 10000	4	4	4	4	4	4	4



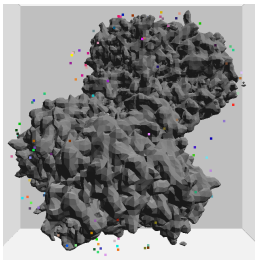
(a) Protein (in wireframe)



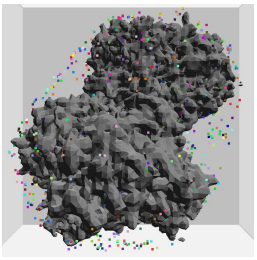
(b) Protein (in spheres)



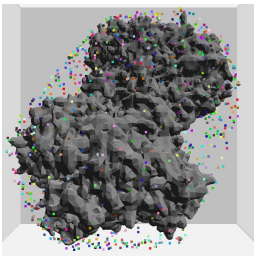
(c) Rigid Protein Obstacle



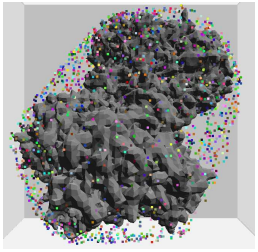
(d) 100 Ligand Samples



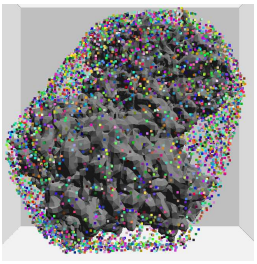
(e) 500 Ligand Samples



(f) 1000 Ligand Samples

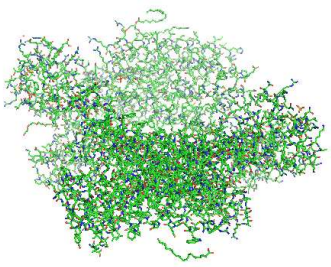


(g) 2000 Ligand Samples

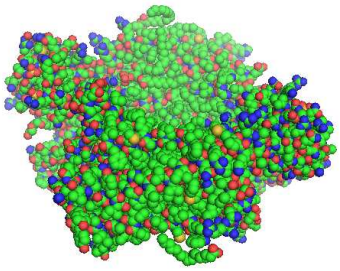


(h) 7000 Ligand Samples

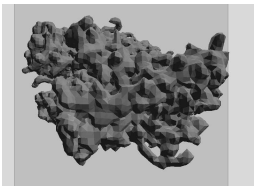
Figure 6.5: Protein 4RRW in wireframe (a) and in spheres (b) viewed by PyMOL [25]. (c) The protein is modeled as a rigid obstacle. (d)-(h) Varying numbers of ligand samples (ligand centers of mass only shown).



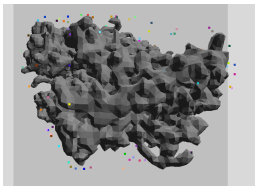
(a) Protein (in wireframe)



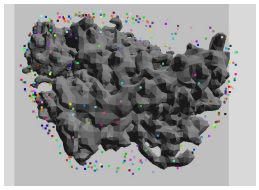
(b) Protein (in spheres)



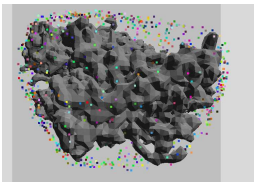
(c) Rigid Protein Obstacle



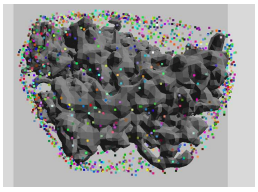
(d) 100 Ligand Samples



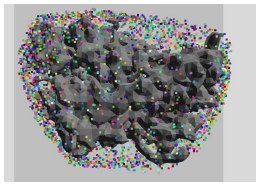
(e) 500 Ligand Samples



(f) 1000 Ligand Samples

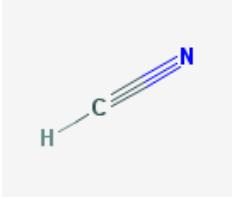


(g) 2000 Ligand Samples

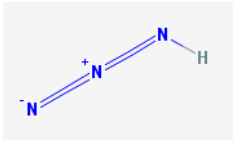


(h) 5000 Ligand Samples

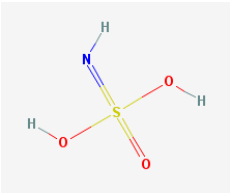
Figure 6.6: Protein 4K5Y in wireframe (a) and in spheres (b) viewed by PyMOL [25]. (c) The protein is modeled as a rigid obstacle. (d)-(h) Varying numbers of ligand samples (ligand centers of mass only shown).



(a) CID: 768

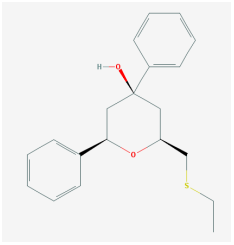


(b) CID: 24530

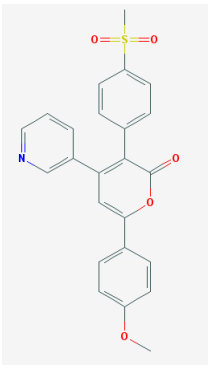


(c) CID: 19366655

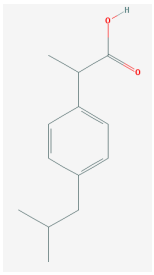
Figure 6.7: Ligand candidates from PubChem [12] for protein 3W6H (see Figure 6.4(a)) ordered by binding affinity rank best to worst.



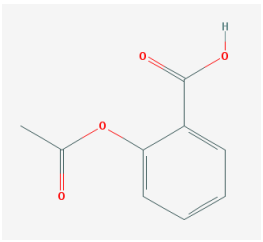
(a) CID: 46224069



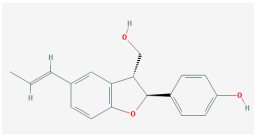
(b) CID: 11441881



(c) CID: 3672

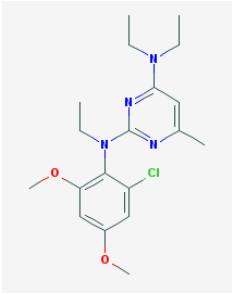


(d) CID: 2244

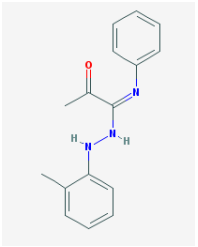


(e) CID: 71460125

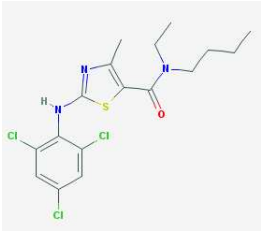
Figure 6.8: Ligand candidates from PubChem [12] for protein 4RRW (see Figure 6.5(a)) ordered by binding affinity rank best to worst.



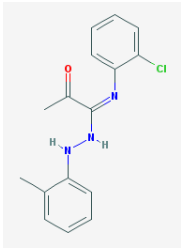
(a) CID: 10595854



(b) CID: 11065415



(c) CID: 10180472



(d) CID:11087955

Figure 6.9: Ligand candidates from PubChem [12] for protein 4K5Y (see Figure 6.6(a)) ordered by binding affinity rank best to worst.

7. CONCLUSION AND FUTURE WORK

In this dissertation we present a novel framework to uniformly sample surfaces in $\mathcal{C}_{space}$. Instead of explicitly constructing the target surfaces, which is generally intractable, our uniform sampling framework only requires detecting intersections between a line segment and the target surface, which can often be done efficiently. Intuitively, since we uniformly distribute the line segments, the intersections between the segments and the surfaces will also be uniformly distributed. We present two instances of our framework, Uniform Obstacle-based PRM (UOBPRM) [22] whose target surfaces are $\mathcal{C}_{obst}$ surfaces, and Uniform Medial-Axis PRM (UMAPRM) [73] whose target surfaces are the medial axis of $\mathcal{C}_{space}$. Sampling on the surface can be difficult since the dimension of a surface is one less than the dimension of the planning space and thus the surface only occupies a small proportion of the entire planning space.

Many motion planning methods have been proposed to sample on surfaces to improve performance or to find high-clearance paths. OBPRM [2] and Gaussian PRM [13] target sampling on obstacle surfaces. Bridge Test PRM [32] was proposed to improve sampling in narrow passages. MAPRM [68, 49] biases sampling towards medial axis surfaces. While some of these methods work well in practice, none of them provides any information regarding the sample distribution on target surfaces. It is useful to know that the surface is sampled with some know distribution, e.g., a uniform distribution, so that one could argue the properties the surface holds such as the probability to plan a solution path on it. The work presented in this dissertation provides for the first time a method for sampling on surfaces with a known distribution, in this case, a uniform distribution.

Our uniform sampling framework works by first uniformly distributing a set of fixed length line segments in $\mathcal{C}_{space}$ and then identifying intersections between line segments and target surfaces (Chapter 3, Section 3.1). We prove that this framework generates configurations uniformly distributed on the target surfaces of $\mathcal{C}_{space}$ (Chapter 3, Section 3.2). Next we provide theoretical guarantees that the uniform sampling framework preserves probabilistic completeness of sampling-based motion planners (Chapter 3, Section 3.3) and demonstrate that its ability to generate samples in the narrow passage is proportional to the surface area of the target surface that bounds the narrow passage (Chapter 3, Section 3.4).

UOBPRM samples $\mathcal{C}_{obst}$ surfaces (Chapter 4, Section 4.1). We evaluated the distribution and efficiency of UOBPRM against other obstacle-based sampling methods and showed that UOBPRM generates more uniformly distributed configurations around $\mathcal{C}_{obst}$ surfaces than other approaches. Moreover, UOBPRM is able to solve some difficult problems more efficiently without computational overhead (Chapter 4, Section 4.3). Finally, we demonstrated that Gaussian PRM is a special case of UOBPRM with particular parameters settings (Chapter 4, Section 4.2).

UMAPRM samples the medial axis of $\mathcal{C}_{free}$ (Chapter 5, Section 5.1). We again evaluated the sample distribution and the efficiency of UMAPRM comparing to MAPRM. We found that UMAPRM configurations are distributed more uniformly along the medial axis and UMAPRM can solve problems that others could not (e.g., a bug trap environment) with negligible computational overhead (Chapter 5, Section 5.2).

In the future, we plan to investigate strategies that balance sample quality and the cost of the node generation, including approaches to tune this for different applications. Since it is difficult to calculate nearest surface witness points in high dimensions, we plan to further explore approximate strategies for applying UMAPRM for

higher dimensional robots. Furthermore, we plan to study if there are other surfaces in $\mathcal{C}_{space}$ on which we could apply this framework in order to improve the quality of node generation in those areas and maybe further generalize to uniformly sample other types of target surfaces.

We used UOBPRM to study the ligand binding affinity ranking problem in computational biology area. By modeling the protein as a rigid obstacle and the ligand as a linkage robot, UOBPRM ligand samples are generated uniformly near protein surfaces which can provide potential ligand/protein binding configurations (Chapter 6, Section 6.2). We analyzed the UOBPRM ligand samples based on affinity metrics to approximate the binding affinity with respect to the protein (Chapter 6, Section 6.2.3). We experimented on three different target proteins and showed our method has potential to rank the ligand binding affinities (Chapter 6, Section 6.3). In future research, we aim to relax our assumption that the protein is rigid and allow some flexing of the protein conformation in response to the ligand. This will be particularly important for protein-ligand complexes that undergo large conformational changes upon binding. We also plan to investigate other affinity metrics including different energy functions.

REFERENCES

[1] N. M. Amato and G. Song. Using motion planning to study protein folding pathways. *J. Comput. Biol.*, 9(2):149–168, 2002. Special issue of Int. Conf. Comput. Molecular Biology (RECOMB) 2001.

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

[3] Nancy M. Amato, O. Burchan Bayazit, Lucia K. Dale, Christopher Jones, and Daniel Vallejo. OBPRM: an obstacle-based PRM for 3d workspaces. In *Proceedings of the third Workshop on the Algorithmic Foundations of Robotics*, pages 155–168, Natick, MA, USA, 1998. A. K. Peters, Ltd. (WAFR '98).

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

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

[6] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Better flocking behaviors using rule-based roadmaps. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 95–111, Dec 2002.

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

[8] Mark L. Benson, Richard D. Smith, Nickolay A. Khazanov, Brandon Dimcheff, John E. Beaver, Peter Dresslar, Jason Nerothin, and Heather A. Carlson. Binding moad, a high-quality protein-ligand database. *Nucleic Acids Research*, 36:674–678, 2008.

[9] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[10] Anna L. Blobaum, Shu Xu, Scott W. Rowlinson, Kelsey C. Duggan, Surajit Banerjee, Shalley N. Kudalkar, William R. Birmingham, Kebreab Ghebreselasie, and Lawrence J. Marnett. Action at a distance: Mutations of peripheral residues transform rapid reversible inhibitors to slow, tight binders of cyclooxygenase-2. *The Journal of biological chemistry*, 290:12793–12803, 2015.

[11] R. Bohlin and L. E. Kavraki. A randomized algorithm for robot path planning based on lazy evaluation. In P. Pardalos, S. Rajasekaran, and J. Rolim, editors, *Handbook on Randomized Computing*, pages 221–249. Kluwer Academic Publishers, 2001.

[12] E. E. Bolton, Y. Wang, P. A. Thiessen, and S. H. Bryant. PubChem: Integrated Platform of Small Molecules and Biological Activities. *Annual Reports in Computational Chemistry*, 4, 2008.

[13] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1018–1023, May 1999.

[14] Kenneth W. Borrelli, Andreas Vitalis, Raul Alcantara, and Victor Guallar. PELE:Protein Energy Landscape Exploration. A Novel Monte Carlo Based Technique. *J. Chem. Theory Comput.*, 1(6):1304–1311, 2005.

[15] Michal Brylinski and Jeffrey Skolnick. A threading-based method (FINDSITE) for ligand-binding site prediction and functional annotation. *PNAS*, 1(105):129–134, 2008.

[16] Michal Brylinski and Jeffrey Skolnick. Comparison of structure-based and threading-based approaches to protein functional annotation. *Proteins*, 1(78):118–134, 2010.

[17] Regina Z. Cer, Uma Mudunuri, Robert M. Stephens, and Frank J. Lebeda. Ic50-to-ki: a web-based tool for converting ic50 to ki values for inhibitors of enzyme activity and ligand binding. *Nucleic Acids Research*, 37:441–445, 2009.

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

[19] Huiling Chen and Huan-Xiang Zhou. Prediction of solvent accessibility and sites of deleterious mutations from protein sequence. *Nucleic Acids Research*, 33(10):3193–3199, 2005.

[20] F. Chiti and C. M. Dobson. Protein misfolding, functional amyloid, and human disease. *Annu. Rev. Biochem.*, 75:333–366, 2006.

[21] C. Chothia. The nature of the accessible and buried surfaces in proteins. *Journal of Molecular Biology*, 105(1):1–12, 1976.

[22] Hsin-Yi Yeh (Cindy), Shawna L. Thomas, David Eppstein, and Nancy M. Amato. UOBPRM: A uniformly distributed obstacle-based PRM. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 2655–2662, 2012.

[23] Juan Cortés, Thierry Siméon, Vicente Ruiz de Angulo, David Guieysse, Magali Remaud-Simon, and Vinh Tran. A path planning approach for computing

large-amplitude motions of flexible molecules. In *ISMB (Supplement of Bioinformatics)*, pages 116–125, 2005.

[24] Juan Cortés, Duc Thanh Le, Romain Iehl, and Thierry Siméon. Simulating ligand-induced conformational changes in proteins using a mechanical disassembly method. *Physical chemistry chemical physics : PCCP*, 12(29):8268–8276, August 2010.

[25] W.L. DeLano. The pymol molecular graphics system (2002). *DeLano Scientific, Palo Alto, CA, USA.*, 2002.

[26] Didier Devaurs, Léa Bouard, Marc Vaisset, Christophe Zanon, Ibrahim Al-Bluwi, Romain Iehl, Thierry Siméon, and Juan Cortés. MoMA-LigPath: A web server to simulate protein-ligand unbinding. *Nucleic Acids Research*, vol. 41:297–302, May 2013.

[27] Raymond N. Dubois, Steven B. Abramson, Leslie Crofford, Rajnish A. Gupta, Lee S. Simon, Leo B. A. Van De Putte, and Peter E. Lipsky. Cyclooxygenase in biology and disease. *FASEB Journal*, pages 1063–1073, 1998.

[28] P. Willett G. Jones and R. C. Glen. Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation. *J. Mol. Biol.*, 245:43–53, 1995.

[29] Yehuda Goldgur, Robert Craigie, Gerson H. Cohen, Tamio Fujiwara, Tomokazu Yoshinaga, Toshio Fujishita, Hirohiko Sugimoto, Takeshi Endo, Hitoshi Murai, and David R. Davies. Structure of the hiv-1 integrase catalytic domain complexed with an inhibitor: A platform for antiviral drug design. *Proc. Natl. Acad. Sci. USA*, 96:13040–13043, 1999.

[30] Christopher Holleman and Lydia E. Kavraki. A framework for using the workspace medial axis in prm planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1408–1413, San Franasisco, CA, 2000.

[31] Kaspar Hollenstein, James Kean, Andrea Bortolato, Robert K. Y. Cheng, Andrew S. Doré, Ali Jazayeri, Robert M. Cooke, Malcolm Weir, and Fiona H. Marshall. Structure of class b gpcr corticotropin-releasing factor receptor 1. *Nature*, 499(7459):438–443, 2013.

[32] D. Hsu, T. Jiang, J.H. Reif, and Z. Sun. Bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4420–4426, 2003.

[33] D. Hsu, J-C. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *Int. J. Robot. Res.*, 25:627–643, July 2006.

[34] D. Hsu, J-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2719–2726, 1997.

[35] D. Hsu, J-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. Comput. Geom. & Appl.*, pages 495–517, 1999.

[36] J. P. Hughes, S. Rees, S. B. Kalindjian, and K. L. Philpott. Principles of Early Drug Discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011.

[37] B Isralewitz, Mu Gao, and K Schulten. Steered molecular dynamics and mechanical functions of proteins. *Current opinion in structural biology*, 11:224–30, 2001.

[38] Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopoly-*

mers, 22(12):2577–2637, December 1983.

[39] E. Katchalski-Katzir, I. Shariv, M. Eisenstein, A.A. Friesem, and C. Aflalo I.A. Vakser. Molecular surface recognition: Determination of geometric fit between proteins and their ligands by correlation techniques. In *Natl. Acad. Sci. USA*, volume 89, pages 2195–2199, 1992.

[40] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.

[41] Y. Koga, K. Kondo, J. Kuffner, and J.C. Latombe. Planning motions with intentions. In *Proc. ACM SIGGRAPH*, pages 395–408, 1995.

[42] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, , and H. Inoue. Motion planning for humanoid robots. In *Proc. 20th Int.l Symp. Robotics Research*, 2003.

[43] KUKA. http://www.youbot-store.com.

[44] I. D. Kuntz, J. M. Blaney, S. J. Oatley, R. Langridge, and T. E. Ferrin. A geometric approach to macromolecule-ligand interactions. *JMB*, 161(2):269–288, 1982.

[45] Eric Larsen, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. Distance queries with rectangular swept sphere volumes. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 4, pages 3719–3726 vol.4, 2000.

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

[47] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *Int. J. Robot. Res.*, 20(5):378–400, May 2001.

[48] M. Levitt. Protein folding by restrained energy minimization and molecular dynamics. *J. Mol. Biol.*, 170:723–764, 1983.

[49] Jyh-Ming Lien, S.L. Thomas, and N.M. Amato. A general framework for sampling on the medial axis of the free space. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 4439–4444, sept. 2003.

[50] Guilin Liu and Jyh-Ming Lien. Fast medial axis approximation via max-margin pushing. In *IROS*, 2015.

[51] T. Liu, Y. Lin, X. Wen, R. N. Jorissen, and M. K. Gilson. BindingDB: a web-accessible database of experimentally determined protein-ligand binding affinities. *Nucleic Acids Res*, 35:198–201, January 2007.

[52] Susanna K. Lüdemann, Valére Lounnas, and Rebecca C. Wade. How do substrates enter and products exit the buried active site of cytochrome p450cam? 1. random expulsion molecular dynamics investigation of ligand access channels and mechanisms. *Journal of Molecular Biology*, 303(5):797–811, 2000.

[53] G.M. Morris, D.S. Goodsell, R.S. Halliday, R. Huey, W.E. Hart, R.K. Belew, and A.J. Olson. Automated docking using a lamarckian genetic algorithm and empirical binding free energy function. *J. Computational Chemistry*, 19:1639–1662, 1998.

[54] Michael M. Mysinger, Michael Carchia, John J. Irwin, and Brian K. Shoichet. Directory of useful decoys, enhanced (dud-e): Better ligands and decoys for better benchmarking. *Journal of Medicinal Chemistry*, 55(14):6582–6594, 2012.

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

Juan, Puerto Rico, October 1979.

[56] Alan R. Saltiel and C. Ronald Kahn. Insulin signalling and the regulation of glucose and lipid metabolism. *Nature*, 414:799–806, 2001.

[57] Amit P. Singh, Jean-Claude Latombe, and Douglas L. Brutlag. A motion planning approach to flexible ligand binding. In *Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 252–261, 1999.

[58] William S. Sly and Peiyi Y. Hu. Human carbonic anhydrases and carbonic anhydrase deficiencies. *Annual Review of Biochemistry*, 64:375–401, 1995.

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

[60] Yousuke Takaoka, Yoshiyuki Kioi, Akira Morito, Junji Otani, Kyohei Arita, Eishi Ashihara, Mariko Ariyoshi, Hidehito Tochio, Masahiro Shirakawa, and Itaru Hamachi. Quantitative comparison of protein dynamics in live cells and in vitro by in-cell 19f-nmr. *Chem. Commun.*, 49:2801–2803, 2013.

[61] Gabriel Tanase, Antal A. Buss, Adam Fidel, Harshvardhan, Ioannis Papadopoulos, Olga Pearce, Timmie G. Smith, Nathan Thomas, Xiabing Xu, Nedal Mourad, Jeremy Vu, Mauro Bianco, Nancy M. Amato, and Lawrence Rauchwerger. The STAPL parallel container framework. In *Proceedings of the 16th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPOPP 2011, San Antonio, TX, USA, February 12-16, 2011*, pages 235–246, 2011.

[62] X. Tang, B. Kirkpatrick, S. Thomas, G. Song, and N. M. Amato. Using motion planning to study RNA folding kinetics. In *Proc. Int. Conf. Comput. Molecular*

Biology (RECOMB), pages 252–261, 2004.

[63] X. Tang, B. Kirkpatrick, S. Thomas, G. Song, and N. M. Amato. Using motion planning to study RNA folding kinetics. *J. Comput. Biol.*, 12(6):862–881, 2005. Special issue of Int. Conf. Comput. Molecular Biology (RECOMB) 2004.

[64] M. Teodoro, G.N. Phillips, Jr, and L.E. Kavraki. Molecular docking: A problem with thousands of degrees of freedom. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 960–965, 2001.

[65] Maxim Totrov and Ruben Abagyan. Derivation of sensitive discrimination potential for virtual ligand screening. In *Third Annual International Conference on Computational Molecular Biology*, pages 312–320, 1999.

[66] Maxim Totrov and Ruben Abagyan. Derivation of sensitive discrimination potential for virtual ligand screening. In *RECOMB*, pages 312–320, 1999.

[67] MN Wass, LA Kelley, and MJE Sternberg. 3dligandsite: predicting ligand-binding sites using similar structures. *NUCLEIC ACIDS RESEARCH*, 38:W469–W473, 2010.

[68] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1024–1031, 1999.

[69] Alexander Wlodawer and Jiri Vondrasek. Inhibitors of hiv-1 protease: a major success of structure-assisted drug design. *Annual Review of Biophysics and Biomolecular Structure*, 27:249–284, 1998.

[70] J. Yang, Ambrish Roy, and Y. Zhang. Biolip: a semi-manually curated database for biologically relevant ligand-protein interactions. *Nucleic Acids Research*, 2012.

[71] Jianyi Yang, Ambrish Roy, and Yang Zhang. Protein-ligand binding site recognition using complementary binding-specific substructure comparison and sequence profile alignment. *Bioinformatics*, 29(20), 2013.

[72] Yuandong Yang and O. Brock. Adapting the sampling distribution in prm planners based on an approximated medial axis. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 5, pages 4405–4410, 2004.

[73] Hsin-Yi (Cindy) Yeh, Jory Denny, Aaron Lindsey, Shawna Thomas, and Nancy M. Amato. UMAPRM: Uniformly sampling the medial axis. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 5798–5803, Hong Kong, P. R. China, June 2014.