WORKSPACE-BASED SAMPLING FOR PROBABILISTIC PATH PLANNING

HANNA KURNIAWATI

A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

2007

Acknowledgement

First of all, I thank my supervisor David Hsu for his guidance and support during my studies. His dedication to research has inspired me to work harder on becoming a better researcher. I am grateful for the time and attention he has generously given.

I thank my thesis comittee Alan Cheng, Leow Wee Kheng, and the external examiner for reading this thesis. Alan has introduced me to David when I was looking for a research supervisor. I am also grateful for his inputs on the WIS strategy. Wee Kheng has provided a space for me in his lab. His advises I often overheard in the lab has broaden my view on research.

In the beginning of my graduate studies, I had a chance to work with Gilardo Sánchez-Ante. I thank him for being an unofficial co-supervisor, a senior, and a colleague, when I need them most. Furthermore, it is always fun to listen to his stories on life and research. I am grateful for the chance to work with Jean-Claude Latombe. His insights and comments have significantly improved this thesis and have encouraged me to dig deeper. I spent a semester in INRIA Rhône-Alpes under the supervision of Thierry Fraichard. The experience has broaden my view on various research in motion planning. I thank Thierry for the opportunity and guidance he has given me while I was in INRIA.

Financial support for my graduate studies have been provided by NUS and INRIA International Internship program.

Most part of this thesis was written in a study carrel of the NUS science library. For that, I am grateful to the librarians and cleaners that have kept the study carrels neat and clean.

In the department, I interacted mostly with Ehsan Rehman, Hendra Setiawan,

Janardan Mishra, Piyush Kanti Bhunre, Saurabh Garg, and Wang Rui Xuan. I thank them for listening to my research problems even though for most of them, my research topic is not even close to theirs. I enjoyed the various discussions we had and I have learnt a lot from them.

Outside the department, I am thankful for the many people that have supported me with their friendship. Fransiska Tanudijaya has helped me settle down when I first arrived in Singapore. Cindy Carmelia's easy going personality and sense of humour always refresh me. Jennifer Tan has become like an older sister to me. Elizabeth has always been there to cheer me up. Julia Ho has offered me many good advice whenever I need one. Bernard and Lilian Lee have impressed me with their hospitality. Lee Hua has helped me find and settle down in a new apartment when I need to move from my old place. Moreover, Bernard and Lilian Lee, Julia Ho, and Lee Hua have enriched my spiritual life. I am also grateful for the friendship of Evelyn Susanto, Fellecia, Hendra Setiawan, Hengky Setiawan, Melina, Sarah, Soffi, Suven, Yenny, and Yi Hui.

I am grateful for the love, care, and support of my family. In particular, my father Djunaidi Slamet Santosa for his constant support and encouragement for me in pursuing the things I like. My mother Inggrianti Subhekti for finally letting me leave home for further study. My aunt Julianti Subhekti that has become like a second mother to me. My brother Satria Kurniawan for introducing me to the interesting world of computer when I was younger and for letting me "mess" around and dominates his computer.

Above all, I thank my heavenly father and savior Jesus Christ for placing all these people in my life so I can learn and grow according to His good purpose. His assurance has kept me going in even the most difficult time.

Contents

Abstract

In this thesis, we show that workspace provides useful information for generating suitable sampling distribution for probabilistic path planning.

Probabilistic path planning has emerged as the most promising approach to path planning of robots with many degrees of freedom. However, it performs poorly when the configuration space contains narrow passages. Furthermore, although understanding the role of probability in probabilistic path planning may lead to the construction of better planners, little work have done so.

We start by presenting an empirical study to explore the role of probability in probabilistic path planning. This study shows that probability is critical to the success of probabilistic path planning, because it relates to the uncertainty that arises from our lack of information about the shape of the robot's configuration space. Sampling distribution represents the planner's belief on the use of sampling from a particular region of the configuration space in improving its understanding about the shape of the robot's configuration space. Furthermore, this study indicates that a suitable sampling distribution depends on the visibility property of the robot's configuration space. It suggests that utilizing workspace information and adapting sampling distribution over time are two potential strategies for improving the current probabilistic planners.

We then present two new workspace-based probabilistic path planners: Workspace Importance Sampling (WIS) and Workspace-based Connectivity Oracle (WCO). Both are based on the multi-query Probabilistic Roadmap (PRM) approach. The novelty of the new planners lies in the sampling strategy.

WIS is a simple workspace-based sampling strategy. It is used mainly as a testbed for exploring the use of workspace information in generating a suitable

sampling distribution. WIS uses workspace information to estimate the size of the visibility set of subsets of the robot's configuration space and constructs sampling distribution based on this estimation. Our experimental results show that this simple sampling strategy is comparable to recent probabilistic path planner that has been shown to perform well. Our analysis shows that the failure probability of WIS converges to zero exponentially in the number of milestones, provided a solution exists. Furthermore, under certain conditions that often happen when the robot moves inside narrow passages in the workspace, WIS has a lower upper bound on its failure probability, compared to the basic-PRM. In addition, we show that in a restricted setting that *does not* depend on the dimensionality of the configuration space, the volume of the visibility sets in the configuration space is bounded by a constant multiplication of the volume of the visibility sets in the workspace. All these strengthen the intuition that workspace provides useful information for generating suitable sampling distributions, and have encouraged us to explore this direction further.

Unlike WIS that generates a static sampling distribution, WCO generates a *dynamic* sampling distribution. It combines workspace information with sampling history and the current state of the roadmap to dynamically adapt its sampling distribution. WCO estimates workspace regions that are more likely to improve the connectivity of the current roadmap, and assigns higher sampling density to subsets of the robot's configuration space that correspond to these workspace regions. Our analysis shows that the failure probability of WCO converges to zero exponentially in the number of milestones, provided a solution exists. Furthermore, when WCO's estimation is "good", the upper bound on the failure probability of WCO is lower than that of the basic-PRM. Our experimental results show that WCO performs up to 28 times faster than the basic-PRM. Furthermore, as the dimensionality of the robot's configuration space increases, WCO's performance decreases slower than other recent probabilistic path planner that has been shown to perform well. We have also successfully implemented WCO in a simulated bridge inspection scenario involving a 35-dofs robot.

List of Tables

List of Figures

Chapter 1

Introduction

To operate in the physical world, robots need to be able to sense, reason, and act. The ability to generate valid motion is crucial for the success of the whole operation. Without this ability, a robot will not be able to perform its task and hence the results of sensing and reasoning will become void. This thesis focuses on efficient algorithms to generate valid motion for various types of robots.

Motion planning addresses the problem of generating valid motion for robots working in a constrained environment. The environment is usually called as the workspace. In the simplest form, a motion planner finds collision-free motion for a robot moving in a perfectly known workspace populated by obstacles. This problem has been proven to be P-space hard [Reif, 1979]. Furthermore, the fastest complete planner to date, *i.e.*, a planner that finds a path whenever one exists and indicates otherwise if none exists, is exponential in the number of degrees of freedom (dofs) of the robot [Canny, 1988]. Unfortunately, many useful robots require high dofs. Commonly used industrial robots have 4-7 dofs and a hyper-redundant robot can have up to hundreds of dofs. In addition to the number of dofs, a robot often has other constraints such as the need to maintain stability while performing its task, the need to avoid collision with itself for hyper-redundant robot, etc. All of these constraints add to the difficulty of motion planning problem.

Despite the hardness of the problem, over the past decade motion planning has emerged as an area of study on its own, solving high-dimensional problems in areas as diverse as robotics [Halperin et al., 1999, Choset et al., 2005], computer

graphics [Koga et al., 1994], computer aided-design [Chang and Li, 1995], and computational biology [LaValle et al., 2000, Amato et al., 2002, Apaydin et al., 2002]. These advances have been made possible with the introduction of probabilistic method [Kavraki et al., 1996b] to motion planning.

Intuitively, a probabilistic planner searches for a path using sampling (a more elaborate explanation is in Section 1.2). As any sampling-based method, the difficulty of finding a solution relies heavily on the relative size of the solution space. If the solution space is small compared to the sampling domain, then searching for a solution is more difficult. Furthermore, as the number of the robot's dofs increases and as more robot's constraints need to be addressed, the size of the solution space compared to the sampling domain tends to decrease. Therefore, to handle more complicated motion planning problems involving higher dofs robot, better sampling strategy for guiding the search becomes more crucial. In response to this, many probabilistic planners with different sampling strategies have been proposed. However, the problem of finding suitable sampling strategies is still largely open.

Moreover, although many probabilistic planners have been proposed, little is known on the role of probability in its success. While this understanding may lead to the construction of better planners, little work have done so.

In this thesis, we propose a possible explanation on the role of probability in probabilistic planning, based on systematic experimental results. With this new understanding, we propose a novel probabilistic planner that uses both workspace information and sampling history to dynamically adapt its sampling distribution.

In this chapter, we will present a short introduction to motion planning and an overview of our main contributions along with an outline of this thesis. We start by presenting an overview of motion planning problems including its variants and various approaches to motion planning in Section 1.1. We then continue with an overview of probabilistic planning approach and the problems with this approach in Section 1.2. Next, we present the problem addressed in this thesis and an overview of our main contrbutions in Section 1.3. Finally, the outline of this thesis is presented in Section 1.4.

1.1 Overview of Motion Planning Problems

Motion planning is a broad problem with many instances. The fundamental problem of all of its instances is to find valid motions for a robotic system to move from one robot's state to another. This problem can be represented in the space of all possible states of the robot as the problem of finding a valid path between two points. A valid path means that all parts of the path lies in the robot's free-space, *i.e.*, the set of all robot's states that does not violate any constraints posed by the motion planning problem at hand. In the rest of this thesis, we use the term robot in a more general sense, to refer to any object that moves in either a 2D or 3D Euclidean space.

1.1.1 Types of Motion Planning Problems

The various instances of motion planning problem can be classified into three types, *i.e.*,

- Path planning problem.

  Path planning is the simplest motion planning problem. It concerns with finding a collision-free motion, between the given initial and goal configurations, for a robot moving in a static workspace populated by obstacles. A robot's configuration is the robot's parameters that define the position of each point on the robot. This problem concerns only with the geometric path of the robot. It assumes that the robot has perfect knowledge about its own geometry and about the workspace's geometry. No errors nor uncertainty about the robot's location or movement are assumed. So, in path planning, the robot's free-space is the set of all configurations that do not cause the robot to collide with any of the obstacles in the environment. Simply put, a path planning problem is a purely geometric problem where all information about the robot and its workspace are a priori known.

  Although this problem seems too simplistic, many real world problems satisfy this requirement. A common example in robotics is planning a rough motion of robots working in an assembly line. In this problem, the geometric

path is needed as input to the robot's controller that will handle the kinematic and dynamic constraints of the robot. In general, the workspace can be considered static because the path planner only needs to consider part of the workspace that is reachable by the robot at the time-frame when the robot needs to move. Moreover since the assembly line in a factory has been designed with high precision such that error is negligible, the perfectly known workspace assumption of path planning is satisfied. Another example is in assembly maintainability study that ensures certain parts of a machine can be easily removed for maintenance purposes [Chang and Li, 1995]. The robot is the machine part. Since the motion for removing the machine part will be carried out by mechanics, we only need to know the geometric path for removing it. In this problem, the workspace is the entire machine. So the workspace is obviously static. In addition, since this study is generally done in a CAD software, it is safe to assume that the error in the workspace model is negligible such that the known workspace assumption of path planning is satisfied.

Moreover in general, planners that work well for solving path planning problems can be extended to handle more complicated motion planning problems. For example, the path planners in [Hsu et al., 1999] and [Kuffner and LaValle, 2000] have been extended successfully to solve more complicated motion planning problems in [Hsu et al., 2002] and [LaValle and Kuffner, 2001], with only little adjustment. In these extensions, both planners consider dynamic workspace and the robot's kinematic and dynamic constraints. The above observations indicate that path planning problem provides a good testbed for developing new motion planners.

In this thesis, we focus mainly on path planning.

- Motion planning with incomplete information.

  In some problems, the workspace may be dynamic. The location of the obstacles may change over time, making time an important parameter in assessing the validity of the robot's motion [Fraichard, 1999]. In other problems, the workspace may only be partially known or even unknown prior to

planning. For instance, a robot explorer may need to generate a map of an old abandoned mine using inputs from its sensors [Thrun et al., 2004]. In this scenario, the motion planner needs to plan the robot's motion such that its sensors can gather the information needed to generate a good map of the mine. Only parts of the workspace that have been captured by the robot's sensors are known to the planner. Futhermore, the workspace information may be inaccurate. Inaccuracies may come due to noise in the sensors or inaccuracies in the robot's movement or the robot's location when sensing took place.

Of course scenarios where the workspace is both dynamic and unknown or only partially known are not rare in real world applications. For instance, a robot acting as a tour-guide in a museum [Thrun et al., 2000] has to deal with dynamic and incomplete workspace information due to the museum's visitors. Since it is infeasible to know the movement of the visitors a priori, the robot only has partial information about the museum, *i.e.*, the geometry of the building and the exhibits. Furthermore, to avoid colliding with the visitors, the robot gather information about its surrounding using its sensors, which in general is noisy. Hence, its information about the dynamic and partially known workspace is also inaccurate.

- Motion planning with differential constraints.

  Some motion planning problems require the planner to generate not just geometric paths, but also the velocity and/or control input for the robot. In this problem, the planner considers not just the robot's geometry, but also the more complicated kinematic and dynamic constraints [Choset et al., 2005]. For instance, certain types of robots, such as a car-like robot, have non-holonomic constraints [Laumond, 1998]. They can only move forward or backward, but not left or right directly.

1.1.2 General Approaches to Motion Planning

Many motion planning methods have been proposed. In the beginning, many works have developed complete motion planners (a summary can be seen in [Latombe,

1991]). These planners return a path whenever one exists and indicates otherwise if none exists. They construct an exact representation of the robot's free-space, which is infeasible for high-dimensional problems. Hence, these planners are only applicable to motion planning involving low-dofs robot, *i.e.*, 2-3 dofs.

Since these complete planners are only applicable for low-dofs robot while many real world applications require robots with more than 3 dofs, several incomplete planners have been proposed. Many of them are based on the potential field approach (a summary can be seen in [Latombe, 1991]). This approach divides the workspace into grids and uses potential function to guide searching for solution. The goal state is represented as an attractive potential where the potential value is set to be minimum. Obstacles are represented as repulsive potentials and the values are set to be infinite. The planner then searches for a solution using gradient descent method, moving towards the minimum of the sum of attractive and repulsive potentials. These planners can get stuck in a local minima.

To avoid problems with local minima, random sampling is introduced [Barraquand and Latombe, 1990]. Initially, random sampling is only used to escape from local minima. Taking a step further, Probabilistic roadmap planning (PRM) [Kavraki et al., 1996b] uses random sampling throughout the entire planning process. It uses uniform random sampling to generate a roadmap that represents the connectivity of the robot's free-space. Sampling enables PRM to avoid the prohibitive cost of generating the exact representation of the robot's free-space. This improved efficiency comes at a cost of completeness. PRM is not complete, but it is probabilistically complete. Probabilistic completeness means that given enough time, the probability that the planner finds a solution whenever one exists converges to one. Although PRM is not a complete planner, experimental results have shown that it is able to solve many motion planning problems involving high-dofs robots working in complicated workspaces [Kavraki et al., 1996b]. The early success of PRM has spurred the development of a new class of motion planners called probabilistic motion planner.

The focus of this thesis is mainly on probabilistic path planning.

1.2 Probabilistic Path Planning

Probabilistic path planning approach aims to generate practical path planning algorithms. Practical means that the planner is able to solve real-world path planning problems within reasonable time. While algorithm means that the planner has some performance guarantee, which in this case is in the form of probabilistic completeness. A planner is probabilistically complete if given enough enough time, the probability that the planner finds a path whenever one exists converges to one.

The main idea of probabilistic path planning is to avoid constructing an explicit representation of the robot's free-space. Instead, it uses *sampling* to construct a graph, called a *roadmap*, as an extreme simplification of the robot's free-space.

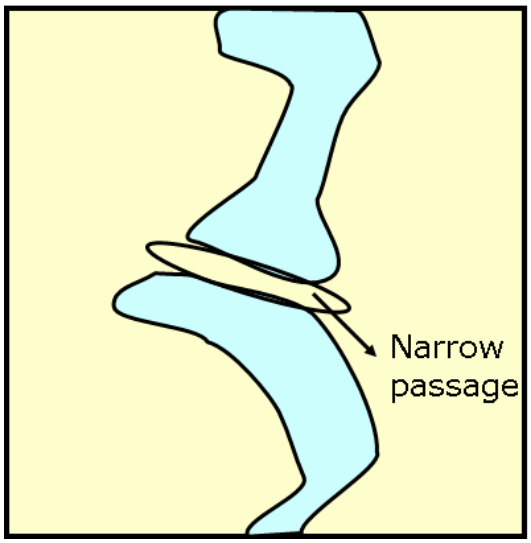


Figure 1.1: An illustration of narrow passage. The robot's free-space is colored light yellow.

Although works in probabilistic path planning (a summary can be seen in [Choset et al., 2005]) have shown the potential of this approach in solving many real-world path planning problems, the performance of probabilistic path planners remain poor when the robot's free-space contains narrow passages. A narrow passage is a small region in the robot's free-space whose removal changes the connectivity of the free-space (an illustration is shown in Figure 1.1). In probabilistic path planning, the problem of capturing the correct connectivity of the robot's free-space when the space contains narrow passages is called the narrow passage problem.

To solve a narrow passage problem, adequate sampling of narrow passages regions is crucial. Since narrow passages have small volume, adequate sampling of narrow passages is difficult, unless the planner is able to identify the location of the narrow passages and then bias sampling towards the narrow passages regions. However, identifying the location of narrow passages is itself a difficult problem

because the planner does not have an explicit representation of the robot's free-space and constructing such an explicit representation requires prohibitive amount of time [Latombe, 1991].

Unfortunately, narrow passages are not rare in real world path planning problems, for example, mobile robots need to pass through narrow corridors or doors while manipulator robots need to insert a peg to a hole. Moreover, due to the curse of dimensionality, as the robot's dofs increases, the size of the narrow passages tend to become smaller and hence the narrow passage problem becomes more severe.

Furthermore, although previous works on probabilistic path planning has generated two types of planners, *i.e.*, multi-query planners and single-query planners, both types of planners face similar difficulties with narrow passages in the robot's free-space [Kavraki et al., 1996b, Cheng et al., 2006]. The two types of planners differ in their objective. The objective of multi-query planner is to quickly answer many queries on the same robot-workspace scenario. It pre-processes the workspace and the robot to generate a graph that represents the robot's free-space reliably. Once the graph is generated, many queries can be answered quickly using efficient graph search method. Single-query planners focus on answering one query as fast as possible, by biasing sampling based on the given initial and goal configurations. This type of planners are useful when only a few queries need to be answered in each robot-workspace scenario. Despite the above differences, the two types of planners use sampling to capture the correct connectivity of the robot's free-space for multi-query planners or to capture the correct connectivity of subset of the robot's free-space that is needed to solve a given query for single-query planners. Hence, both types of planners face similar difficulties when the robot's free-space contains narrow passages.

1.3 Problem Scope and Main Contributions

This thesis focuses on the narrow passage problem in multi-query probabilistic path planning. In particular, the goal of this thesis is to find better sampling strategies to improve the performance of current multi-query probabilistic path

planners in solving narrow passage problems.

Although many sampling strategies [Amato et al., 1998,Boor et al., 1999,Cheng et al., 2006,Foskey et al., 2001,Holleman and Kavraki, 2000,Sun et al., 2005] have been proposed to improve probabilistic path planning in solving narrow passage problem, the narrow passage problem remains open. Furthermore, despite the many sampling strategies that have been proposed, little is known of the role of probability in probabilistic path planning. While understanding the role of probability in probabilistic path planning may give hints on how to generate a more suitable sampling strategy, prior works on probabilistic path planning tend to take probability for granted.

In this thesis, we start by performing an empirical study to understand the role of probability in probabilistic path planning. We articulated the distinction between two main components of random sampling, *i.e.*, sampling distribution and sampling source, which has been blurred in previous works on probabilistic path planning. By articulating this distinction, we are able to explore ways on improving current probabilistic path planners in a more structured manner. This distinction and our empirical study clarify the importance of sampling distribution to the overall performance of probabilistic path planning. Our study indicates that suitable sampling distribution depends on the visibility property of the robot's free-space. Furthermore, it suggests that utilizing workspace information and generating dynamic sampling distribution are two potential avenues for improving current probabilistic path planning in solving narrow passage problem.

Utilizing the results of our empirical study, we propose two new probabilistic path planning, called Workspace Importance Sampling (WIS) and Workspace-based Connectivity Oracle (WCO). Both planners exploit workspace information to generate a more suitable sampling distribution.

WIS is a simple workspace-based sampling strategy. It is used mainly as a tool to explore the use of workspace information in generating suitable sampling distribution for probabilistic path planning. Intuitively, a narrow passage in the robot's free-space means that the robot is surrounded by obstacles that lie close to each other in the workspace, such that a little displacement of the robot causes it

to collide with one or more obstacles in the workspace. So to bias sampling towards narrow passages regions, WIS uses distance between obstacles in the workspace to construct a static sampling distribution. Our experimental results show that this simple sampling strategy is comparable to recent probabilistic path planner that has been shown to perform well. Our analysis shows that WIS is probabilistically complete. Furthermore, under certain conditions that often happen when the robot moves inside a narrow region in the workspace, WIS has a lower upper bound on its failure probability, compared to the basic-PRM.

To understand the use of workspace information further, we explore the relation between the visibility property of the robot's free-space and the workspace. In this thesis, we show that under certain conditions that *do not* depend on the dimensionality of the robot's free-space, the visibility sets of configurations in the robot's free-space are "bounded" by the visibility sets of the corresponding points in the workspace. This relation indicates that workspace information can be used to estimate the visibility property of the robot's free-space. Since our initial empirical study has indicated that suitable sampling distributions depend on the visibility property of the robot's free-space, workspace information can potentially be used for generating suitable sampling distributions for probabilistic path planning.

WCO uses workspace information along with sampling history and the current state of the roadmap to generate a *dynamic* sampling distribution. Workspace information provides a rough global view of the connectivity of the robot's free-space. While sampling history provides local information about the geometry of the robot's free-space. Combining this two complementary information enable WCO to generate a more suitable sampling distribution for the problem at hand. Furthermore, dynamic sampling distribution allows the planner not to waste resources for over-sampling well represented regions of the robot's free-space. WCO consists of multiple component samplers. Each sampler estimates workspace regions that are more likely to improve the connectivity of the current roadmap and assigns higher sampling density to subsets of the robot's configuration space that correspond to these workspace regions. Sampling history is then used to favor samplers that have been performing well in the past. Our analysis shows that WCO

is probabilistically complete. Furthermore, when its estimation satisfies a certain criteria, the upper bound on the failure probability of WCO is lower than that of the basic-PRM. Our experimental results show that WCO performs significantly faster than recent probabilistic path planner that has been shown to perform well. Furthermore, as the dimensionality of the robot's configuration space increases, WCO's performance decreases slower than other recent probabilistic path planner. We have also successfully implemented WCO in a simulated bridge inspection scenario involving a 35-dofs robot.

In short, this thesis shows that workspace provides useful information for generating suitable sampling distributions, which is critical for improving the performance of probabilistic path planning in narrow passage problem. The main contributions of this thesis are as follows,

1. Articulation of the distinction between two main components of random sampling, *i.e.*, sampling distribution and sampling source, in probabilistic path planning.

2. Effective method for exploiting workspace information which leads to new probabilistic path planners that significantly improve the performance of recent probabilistic path planner in solving narrow passage problems.

1.4 Outline

This thesis shows that workspace provides useful information for generating suitable sampling distributions, which then speed-up current probabilistic path planners in solving narrow passage problems.

We start by presenting our exploration on ways to improve current probabilistic path planning in solving the narrow passage problem in Chapter 2. In this chapter, we articulate the distinction between two main components of random sampling, *i.e.*, sampling distribution and sampling source, in probabilistic path planning. This articulation enables us to explore strategies for improving probabilistic path planning in solving the narrow passage problem in a more systematic way. We then present our hypothesis on the role of sampling distribution in probabilistic path

planning and present an empirical study that shows the importance of sampling distribution in probabilistic path planning. Furthermore, this study indicates that suitable sampling distribution depends on the visibility property of the robot's free-space. We end this chapter by suggesting that utilizing workspace information and generating dynamic sampling distribution are two possible avenues for improving current probabilistic path planners in solving narrow passage problems.

We continue by exploring the use of workspace information to generate suitable sampling distributions for probabilistic path planing in Chapter 3. For this, we explore the relation between the visibility property of the robot's free-space and that of the workspace. And then present a simple workspace-based sampling strategy, called *Workspace Importance Sampling (WIS)*.

Next in Chapter 4, we present the core of this thesis, a new probabilistic path planner called *Workspace-based Connectivity Oracle (WCO)* that uses *workspace information* to generate *dynamic sampling distribution*. WCO consists of multiple samplers where each sampler uses workspace information to generate a dynamic sampling distribution. The multiple samplers are combined using adaptive hybrid sampling approach. So to understand WCO better, we further explore adaptive hybrid sampling approach in Chapter 5.

Finally, we present the conclusion and possible future work of this thesis in Chapter 6.

Chapter 2

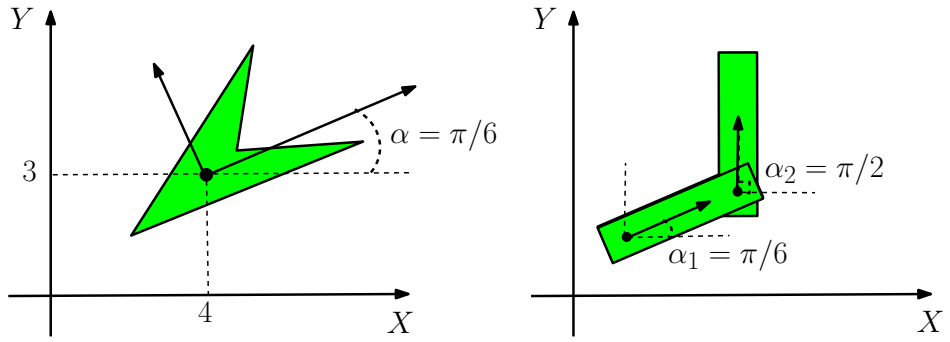
The Role of Probability in Probabilistic Path Planning

In this chapter, we present an empirical study on the importance of sampling distribution in probabilistic path planning. We articulate the distinction between sampling distribution and sampling source, which has been blurred in probabilistic path planning literature. By doing so, we are able to clarify the importance of sampling distribution in probabilistic path planning. This study indicates that a suitable sampling distribution is critical for the success of probabilistic path planning because it represents the uncertainty that arises from our lack of information on the shape of the robot's configuration space. Furthermore, this study suggests that utilizing workspace information and generating dynamic sampling distributions are two possible avenues for improving current probabilistic path planners in solving narrow passage problem.

We start with a short description of configuration space, *i.e.*, a framework to represent path planning problem, in Section 2.1. This framework is crucial for the development of probabilistic path planners. We then continue with an overview of probabilistic path planning in Section 2.2. In Section 2.3, we present our hypothesis on the role of probability in probabilistic path planning. We then present systematic experimental results that show the importance of sampling distribution in probabilistic path planning in Section 2.4. Next, we present a guideline on what the desired sampling distribution is in Section 2.5 and a literature review on

various sampling strategies for probabilistic path planning in Section 2.6. Finally, we end this chapter by presenting possible directions to improve probabilistic path planning in solving narrow passage problem.

2.1 Configuration Space



(a) Rigid body robot. (b) Articulated robot.

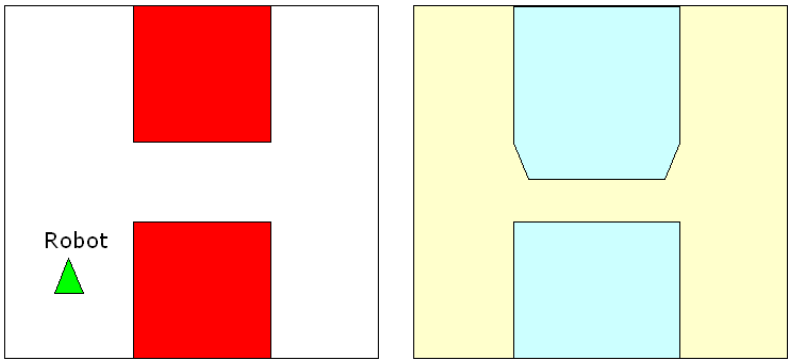
Figure 2.1: Illustration of a robot's configuration. The black dot is the origin of the frame attached to the the rigid body(ies) that constructs the robot.

A configuration of a robot is a set of parameter values that specifies the position of every point of the robot in the workspace. For example, for a polygonal rigid body robot in Figure 2.1(a), $(x = 4, y = 3, \alpha = \pi/6)$ is a configuration of the robot. While for an articulated robot in Figure 2.1(b), $(\alpha_1 = \pi/6, \alpha_2 = \pi/2)$ is a configuration of the robot.

Configuration space $\mathcal{C}$ of a robot is then the set of all possible configurations of the robot. The dimension of $\mathcal{C}$, denoted as $\dim(\mathcal{C})$, is the same as the number of degrees of freedom (dofs) of the robot, and is defined as the minimum number of parameters that uniquely specifies each configuration of the robot. A robot's configuration may have many parameterization. For example, a polygonal rigid body robot can be parameterized by the (x, y)-coordinate of its center of mass and the orientation, or by the (x, y)-coordinates of two different points on the robot. However, these are just different representations of the same abstract $\mathcal{C}$. The dimension of $\mathcal{C}$ is computed based on the number of parameters in the parameterization that uses the least number of parameters. So, in the above example, regardless of the

parameterization used, the robot has 3-dofs and $\dim(\mathcal{C}) = 3$.

The configuration space $\mathcal{C}$ consists of two subsets, *i.e.*, the free-space $\mathcal{F}$ and its complement the forbidden region $\mathcal{C}\backslash\mathcal{F}$. The free-space is the set of configurations that place the robot such that it does not collide with any obstacles in the workspace. In other words, the configurations in $\mathcal{F}$ place each point of the robot in the workspace free-space $\mathcal{W}_{\mathcal{F}}$, *i.e.*, regions of the workspace that are not occupied by any of the obstacles. So in $\mathcal{C}$, the path planning problem, *i.e.*, finding a collision-free path for a robotic system working in a workspace occupied by obstacles, is reduced to finding a path for a point in $\mathcal{F}$.



(a) (b)

Figure 2.2: Illustration of a robot's configuration space. (a) The robot in its workspace. The robot is the green triangle with the vertex in bottom left as the origin of the robot's frame. It can only translate. The obstacles are the red rectangles. (b) The corresponding configuration space $\mathcal{C}$. The robot's free-space $\mathcal{F}$ is colored light yellow while the forbidden region $\mathcal{C}\backslash\mathcal{F}$ is colored light green.

Although for a simple robot, constructing the exact representation of $\mathcal{F}$ can be done efficiently by Minkowski difference computation (see Figure 2.2 for an example), for a more general robot, constructing an exact representation of $\mathcal{F}$ may be impractical. For instance, if we slightly increase the complexity of the polygonal robot in Figure 2.2(a) such that it can translate and rotate, we need to discretize the angle and then construct the Minkowski difference for each angle value. As expected, construction of an exact representation of $\mathcal{F}$ becomes even more complicated and takes prohibitive amount of time as $\dim(\mathcal{C})$ increases.

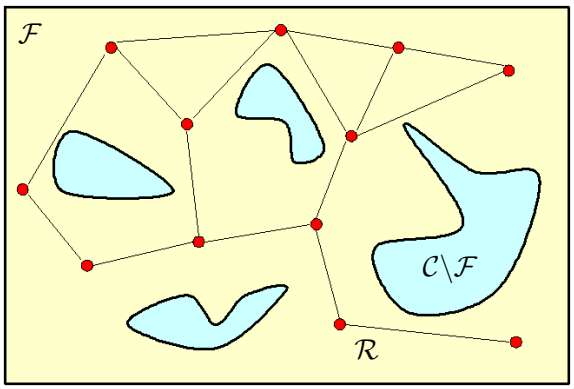


Figure 2.3: The roadmap $\mathcal{R}$. The milestones are shown as small red disks. Two milestones are connected by an edge whenever the straight-line segment connecting the two milestones are collision-free. The robot's free-space $\mathcal{F}$ is colored light yellow while the forbidden region $\mathcal{C}\backslash\mathcal{F}$ is colored light green.

2.2 Overview of Probabilistic Path Planning

The main idea of probabilistic path planning [Kavraki et al., 1996b] is to avoid constructing an exact representation of $\mathcal{F}$, by constructing a graph, called roadmap $\mathcal{R}$, which is a 1-dimensional representation of the connectivity of $\mathcal{F}$. The nodes in $\mathcal{R}$ are configurations $q \in \mathcal{F}$ and are often called milestones. The edges are paths in $\mathcal{F}$ that connect two milestones together.

To generate milestones and edges of $\mathcal{R}$ without constructing an exact representation of $\mathcal{F}$, a probabilistic planner uses two inexpensive primitives, *i.e.*, `FreeConf`(q) and `FreePath`(q, q'), for probing the geometry of $\mathcal{F}$. The first primitive `FreeConf`(q) returns true if and only if $q \in \mathcal{F}$. It checks whether the robot placed at q collides with any of the obstacles in the workspace. The second primitive `FreePath`$(q,\ q')$ returns true if and only if the edge between q and q' is collision-free, *i.e.*, the straight line segment $\overline{qq'} \in \mathcal{F}$. It checks whether the robot can move according to the configurations in $\overline{qq'}$ without colliding with any of the obstacles in the workspace. A survey of exact algorithms that can perform the two primitives fast, even for workspace with large and complicated geometry, can be found in [Lin and Manocha, 2004].

To construct $\mathcal{R}$, the planner samples the robot's free-space $\mathcal{F}$ according to a particular distribution. To generate samples in $\mathcal{F}$, the planner samples configurations from $\mathcal{C}$ and then use `FreeConf` to check whether the sampled configurations

are collision-free or not. Every time a collision-free configuration is sampled, the planner inserts the configuration to the current roadmap $\mathcal{R}$. To insert a configuration to $\mathcal{R}$, it first adds the new configuration as a milestone in $\mathcal{R}$. Then, it tries to connect the new milestone to other existing milestones in $\mathcal{R}$ to generate the edges of $\mathcal{R}$. An overall algorithm is shown in Algorithm 2.1.

The program receives a set of queries Q as input. Each element in Q is a query represented as a 2-tuple $< q_i, q_g >$ of initial q_i and goal q_g configurations. Furthermore, we set the maximum number of milestones in a roadmap to avoid the planner from running forever, because probabilistic planners cannot tell whether a given query is actually solvable or not. The program stops whenever all queries have been solved or the number of milestones in the roadmap has reached the maximum. The procedure `AllSolved` tests whether all the given queries have been solved. It returns a list of paths Ψ where each element of Ψ is a solution to a query configuration.

The procedure `Sample` samples a configuration from $\mathcal{C}$. The basic-PRM [Kavraki et al., 1996b] uses uniform random sampling over $\mathcal{C}$ and as we will soon see in Section 2.6, there can be many ways to sample $\mathcal{C}$. The main objective of the various sampling strategies that have been proposed is to improve the performance of probabilistic path planning in solving narrow passage problem.

Despite the success of probabilistic path planning in solving many real-world path planning problems (*e.g.*, [Chang and Li, 1995, Kavraki et al., 1996b, Koga et al., 1994]), narrow passage problem remains open. As discussed in Section 1.2, narrow passage problem is the problem of capturing the correct connectivity of the robot's free-space $\mathcal{F}$ when $\mathcal{F}$ contains narrow passages, *i.e.*, small regions whose removal changes the connectivity of $\mathcal{F}$. Previous works have formally articulated narrow passage in terms of the visibility property of $\mathcal{F}$ (a short overview is presented in Section 2.5). A configuration $q' \in \mathcal{F}$ is visible from another configuration $q \in \mathcal{F}$ whenever the straight line segment $\overline{qq'}$ between q and q' lies entirely in $\mathcal{F}$. And a subset $\mathcal{F}_1$ of $\mathcal{F}$ is visible from another subset $\mathcal{F}_2$ of $\mathcal{F}$ whenever each configuration in $\mathcal{F}_1$ is visible from each configuration in $\mathcal{F}_2$. Intuitively, narrow passage problem happens when a large subset of a path-connected component $\mathcal{F}'$ of $\mathcal{F}$ is

Algorithm 2.1 Probabilistic path planning(Q, N, D, maxNeigh)

repeat
  `Sample`(q)
  if `FreeConf`(q) returns true **then**
    Add q as a new milestone in $\mathcal{R}$.
    $Neigh_m \leftarrow$ sorted list of milestones in $\mathcal{R}$ within D distance from m, sorted ascendingly based on the distance to m.
    nNeigh = 0
    i = 1
    while nNeigh < maxNeigh and $Neigh_m[i]$ is not end-of-list **do**
      if $Neigh_m[i]$ and m belong to different connected components of $\mathcal{R}$ **then**
        if `FreePath`($Neigh_m[i]$, m) returns true **then**
          Add an edge between $Neigh_m[i]$ and m in $\mathcal{R}$.
        Increment nNeigh by 1.
      Increment i by 1.
until `AllSolved`(Q, $\mathcal{R}$, Ψ) is true or $\mathcal{R}$ contains N milestones.
Return Ψ.

visible only from a small region of $\mathcal{F}'$.

Solving narrow passage problem within reasonable time is crucial, as this problem often occurs in real-world path planning problems and the problem tends to become more severe as the number of dofs of the robot increases. In the rest of this chapter, we explore the role and importance of sampling distribution in order to find potential avenues for generating a more suitable sampling strategy to improve the performance of probabilistic path planning in solving narrow passage problem.

2.3 The Role of Probability in Probabilistic Path Planning

Although many sampling strategies have been proposed and previous studies have shown that different sampling strategies may drastically change the performance of probabilistic path planning, it is not clear why. It is not even clear why probability is needed in probabilistic path planning, considering that there are no inherent uncertainty in a path planning problem. Furthermore, although understanding the role of sampling distribution may give hints on how to improve probabilistic path planning further, little work has done so. In this section, we present a possible

explanation on the role of sampling distribution in probabilistic path planning.

Despite the absence of inherent uncertainty in path planning problems, probabilistic planners never know the exact shape of $\mathcal{F}$. It implicitly maintains many hypotheses about the shape of $\mathcal{F}$ that are consistent with the current roadmap. Suppose $\mathcal{H}$ is the set of all consistent hypotheses. Constructing a roadmap is then the same as pruning out wrong hypotheses from $\mathcal{H}$ until the ones left behind can solve the given queries. Each sampled configuration reveals some information about $\mathcal{F}$ and hence prunes some wrong hypotheses from $\mathcal{H}$. Different sample prunes different subset of $\mathcal{H}$. And the amount of queries that can be solved because of the pruning may differ, too. Obviously, it is preferable to sample configurations such that the planner quickly reaches the set of hypotheses that can solve the given queries. Probabilistic path planning uses probability to reflect its belief on how useful the pruning generated by the configurations are. Hence, probability in probabilistic path planning is used to represent the uncertainty that comes from the planner's lack of information about the exact shape of $\mathcal{F}$.

Let's now make the notion of hypothesis more concrete. A hypothesis H on the shape of $\mathcal{F}$ is a binary relation over $\mathcal{F}$ where $(q, q') \in H$ means that according to the hypothesis, configurations q and q' lie in the same connected component of $\mathcal{F}$. A roadmap $\mathcal{R}$ represents a binary relation over $\mathcal{F}$, too. Let's denote the binary relation represented by $\mathcal{R}$ as R. Then, $(q, q') \in R$ means that there exists milestones m and m' in $\mathcal{R}$ such that q and q' lie in the visibility set of m and m' respectively (*i.e.*, the straight line segments $\overline{qm}$ and $\overline{q'm'}$ lie entirely in $\mathcal{F}$), and both m and m' belong to the same connected component of $\mathcal{R}$. We can then define a consistent hypothesis as follows,

Definition 2.1 *A hypothesis H is consistent with roadmap $\mathcal{R}$ that represents binary relation R whenever $R \subseteq H$.*

Let's denote the set of all consistent hypotheses as $\mathcal{H}$. In terms of hypothesis, a query between two configurations q and q' is solved whenever all hypotheses in $\mathcal{H}$ agrees, *i.e.*, all hypotheses in $\mathcal{H}$ contains (q, q'). At any moment during planning, there are many consistent hypotheses. For instance, at the beginning when no milestone has been inserted to $\mathcal{R}$, any hypothesis is consistent with $\mathcal{R}$.

As roadmap construction progresses, more information about $\mathcal{F}$ are known, less hypotheses are consistent with the roadmap, and more queries can be solved. The set of queries `solved(`$\mathcal{H}$`)` that can be solved by $\mathcal{H}$ can then be defined as follows,

Definition 2.2 *Suppose $\mathcal{H}$ is the set of all hypotheses consistent with roadmap $\mathcal{R}$ and R is the binary relation represented by $\mathcal{R}$. Then, the set of queries* `solved(`$\mathcal{H}$`)` *that can be solved using $\mathcal{H}$ is* `solved(`$\mathcal{H}$`)` $= \bigcap_{\forall H \in \mathcal{H}} H = R$.

Generating a good roadmap fast means that one would like to quickly converge to $\mathcal{H}$ with the largest set of solvable queries. Ideally, one would like to quickly find the smallest set of milestones that generates a roadmap where its corresponding set of consistent hypotheses $\mathcal{H}$ has the largest set of solvable queries. However, since the exact $\mathcal{F}$ is unknown, probabilistic path planning uses sampling distribution to represent its belief on which configurations are more likely to cause fast convergence.

2.4 How Important Sampling Distribution is?

Now that we have a possible explanation on the role of sampling distribution in probabilistic path planning, the question remains as to how important is sampling distribution in probabilistic path planning? To answer this question, we first clarify the distinction between two main components of random sampling, *i.e.*, sampling distribution Pr and sampling source Sr. Sampling distribution is the probability used for sampling points from the configuration space $\mathcal{C}$. For instance, the sampling distribution of the basic-PRM is uniform, which means the planner samples points from every parts of $\mathcal{C}$ equally likely. Sampling source Sr specifies how to generate uniformly distributed random number, which will then be used to generate random samples according to a particular distribution Pr. For instance, the sampling source in the basic-PRM is pseudo-random number generator. Although the distinction between sampling distribution and sampling source seems obvious, previous works on probabilistic path planning have blurred this distinction.

By articulating the distinction between sampling distribution and sampling source, it becomes clear that recent work [LaValle et al., 2004], that has started to

become a trend in probabilistic path planning, argues that by modifying sampling source, we can significantly improve the overall performance of the planner. They argue for the use of quasi-random number generator, instead of pseudo-random number generator, as the sampling source. We will present a more elaborate explanation and a short survey on quasi-random number generator and sampling strategies that use quasi-random number generator in the next section. But before that, to assess the importance of sampling distribution, below we present an empirical study that compares the effect of improving sampling distribution and improving sampling source to the overall performance of the planner.

The implementation details and experimental setup of our empirical study are presented in Section 2.4.1. The results and discussion are presented in Section 2.4.2. To ease later discussions, let's denote a sampling strategy as a pair of sampling distribution and sampling source (Pr, Sr).

2.4.1 Implementation Details and Experimental Setup

		Sampling source (Sr)		
		Pseudo-random number generator (ran)	Quasi-random number generator	
			Halton sequence (hal)	Incremental discrepancy optimal (opt)
Sampling distribution (Pr)	Uniform (U)	(U, ran)	(U, hal)	(U, opt)
	Gaussian (G)	(G, ran)	(G, hal)	(G, opt)

Figure 2.4: The sampling strategies used for comparing the effect of sampling distribution and sampling source to the overall performance of the planner.

To assess the significance of sampling distribution and sampling source in probabilistic path planning, we compare the performance of six different sampling strategies in solving narrow passage problem. The six sampling strategies are combinations of two different sampling distributions and three different sampling sources (Figure 2.4 illustrates the combination). For sampling distribution, we use uniform distribution [Kavraki et al., 1996b] and Gaussian strategy [Boor et al., 1999]. Uniform distribution, denoted as U, is used as a comparator. While Gaus-

sian strategy, denoted as G, is used because of its simplicity. In short, Gaussian strategy assigns higher sampling density to configurations near the boundary of the free-space. It samples a configuration in two stages. First, it samples a collision configuration q. Then it samples another configuration q', where the distance between q and q' is sampled based on a Gaussian distribution centered at q. A more elaborate explanation of Gaussian strategy is presented in Section 2.6.1. For sampling source, we use pseudo-random number, Halton sequence [Niederreiter, 1992], and incremental discrepancy-optimal sequence [Lindemann and LaValle, 2003] to represent the various sampling sources Sr. Pseudo-random number, denoted as *ran*, is used as a comparator. While Halton sequence, denoted as *hal*, and incremental discrepancy-optimal, denoted as *opt*, are used because they have been shown to perform well and better than other quasi-random sequence in solving path planning problems [LaValle et al., 2004]. A more elaborate explanation on these sampling sources are presented in Section 2.6.2. To assess the effect of sampling distribution and sampling source to the overall performance of the planner, we will compare the performance of the six combinations of (U, *ran*), (G, *ran*), (U, *hal*), (G, *hal*), (U, *opt*), and (G, *opt*) on several path planning problems that require the robot to pass through one or more narrow passages.

The planner and all its sampling strategies are implemented using C++, based on the algorithm presented in Algorithm 2.1. Our implementation assumes that the workspace and $\mathcal{C}$ are normalized. The pseudo-random number generator is based on the code provided in `http://www-cs-faculty.stanford.edu/~knuth/programs/rng-double.c`, while the quasi-random number uses the code provided in `http://msl.cs.uiuc.edu/~slindema/sampling/`. Implementation details of the primitives, *i.e.*, `FreeConf`, `FreePath`, and `AllSolved` are presented in Appendix A.

The implementation of Gaussian strategy with quasi-random source, *i.e.*, (G, *hal*) and (G, *opt*), needs more elaboration to ensure that we do not introduce additional bias due to the use of two random numbers for every sample of Gaussian strategy. Note that there has not been any prior works on non-uniform measure, including Gaussian strategy, with quasi-random source before. To reduce the pos-

sibilities of additional bias, for the experiments in this chapter, we slightly modify the Gaussian strategy presented in [Boor et al., 1999] such that the second configuration is sampled independently for each dimension. In other chapters, we implemented Gaussian strategy as presented in [Boor et al., 1999]. The algorithm for Gaussian strategy used in this chapter is shown in Algorithm 2.2. We implement two independent copies of the deterministic source, one for sampling q (step 2 of Algorithm 2.2) and the other for sampling q' (step 4 of Algorithm 2.2). Each of them advances independently. As an illustration, suppose we want to get 2 milestones and suppose $Sr_1 = \{sr_1^1, sr_1^2, sr_1^3 \ldots\}$ and $Sr_2 = \{sr_2^1, sr_2^2, sr_2^3 \ldots\}$ are 2 independent sequences of the deterministic source. To get a milestone, we need to sample an in-collision configuration for q and a collision-free configuration for q'. We use Sr_1 for sampling q and Sr_2 for sampling q'. For the first milestone, we sample q using Sr_1 starting from sr_1^1 and q' using Sr_2 starting from sr_2^1, until we get q and q' such that q is in-collision and q' is collision-free. Note that in general, the index in Sr_1 advances faster than that in Sr_2 because we will not sample q' unless the sampled q is in-collision (see Algorithm 2.2). Suppose we get the first milestone when q is sr_1^8 and q' is $f(s_2^4)$ where f is a transformation that transforms uniform distribution to Gaussian distribution. To sample the second milestone, we follow the same procedure and continue using Sr_1 and Sr_2 to sample q and q' respectively. So, the only difference is that we use Sr_1 starting from sr_1^9 instead of sr_1^1 and we use Sr_2 starting from sr_2^5 instead of sr_2^1. In transforming deterministic random numbers from uniform distribution to Gaussian distribution, we need to preserve discrepancy [Niederreiter, 1992]. Therefore, we use Moro's inversion [Moro, 1995] instead of the commonly-used Box-Mueller conversion.

Algorithm 2.2 Gaussian Strategy(σ).

1: **loop**
2: q = a random configuration from a uniform distribution.
3: **if** `FreeConf`(q) returns false **then**
4: q' = a random configuration where each dimension-i of q' is sampled from a Gaussian distribution with mean $q[i]$ and standard deviation σ.
5: **if** `FreeConf`(q') returns true **then**
6: Save q' as a milestone in the roadmap.

We use the Euclidean distance metric in all our experiments. One may argue

that different distance metric may generate different results as one of the proposed criteria, *i.e.*, dispersion, for good sampling source (surveyed in Section 2.6.2) is based on distance metric. However in terms of sampling, the idea of modifying the distance metric is the same as the idea of modifying the sampling distribution. Both try to bias sampling towards regions where valid configurations are more difficult to sample. This idea is quite different from the idea of modifying the sampling source, which is to spread the samples as "evenly" as possible in the space.

For strategies with pseudo-random source, the results in each scenario are averaged over 30 independent runs. For strategies with quasi-random source, we only need to run once for each scenario because the results are deterministic. Each run were terminated once all the given queries have been solved. The experiments were conducted on a PC with Intel Pentium 4 processor 3GHz and 1GB RAM.

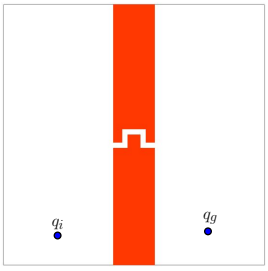
2.4.2 Experimental Results

One main objective in improving probabilistic planning is to improve its performance in solving narrow passage problem and to increase the number of dimensions a motion planner can handle well.

As the visibility property of $\mathcal{F}$ worsen

To assess the significance of sampling distribution and sampling source in solving narrow passage problems, we test how changing the measure and the source affect the planner's performance as the narrow passage problem becomes more severe. For this, we construct an environment with varying size of narrow opening. When the opening becomes narrower, the visibility set of the points inside the passage becomes smaller. And hence the visibility property of $\mathcal{F}$ worsen.

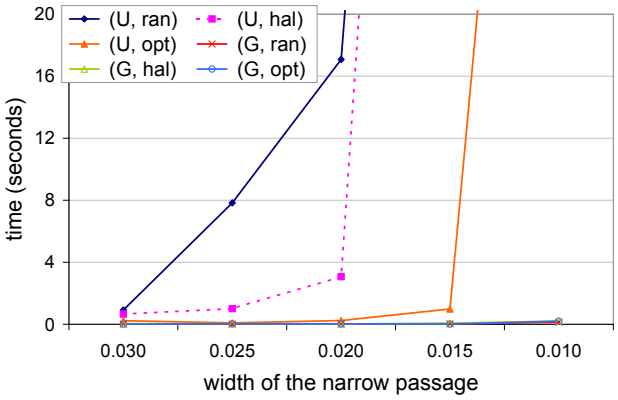
We compare the six sampling strategies on the scenarios shown in the left-most side of Figure 2.5. In scene-1, the robot is a point robot. It has to pass through the narrow corridor in-between the two obstacles in order to solve the given query. To vary the visibility property of $\mathcal{F}$, we use varying corridor width, *i.e.*, 0.03, 0.025, 0.02, 0.015, and 0.01. The results are beside scene-1 in Figure 2.5. The table



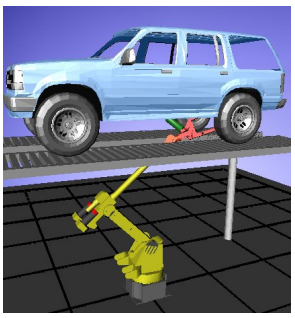
scene-1

	U	G
ran	1.0	92
hal	1.4	23
opt	4.0	92

results of scene-1 for corridor width 0.03



results of scene-1 as the corridor width decreases



scene-2

	U	G
ran	1.0	40.3
hal	3.9	33.2
opt	0.9	42.2

results of scene-2

Figure 2.5: Scenarios and results of sampling distribution vs sampling source in solving narrow passage problem.

shows the speed-up of the different sampling strategies over (U, ran) in building a roadmap that can answer the given query when the corridor width is 0.03. And the chart shows the time required as the corridor width decreases. For strategies with pseudo-random number generator as the sampling source, the results are the average results of the 30 runs.

The three indistinguishable curves bundled together at the bottom of the plot correspond to Gaussian strategies with various sampling sources. The results show that as the narrow passage problem becomes more severe, improving the sampling distribution generates a much higher improvement on the overall performance of the planner, compared to improving the sampling source. This is reasonable because as the narrow passage problem becomes more severe, in general more milestones are needed to solve the problem. When the number of milestones is large,

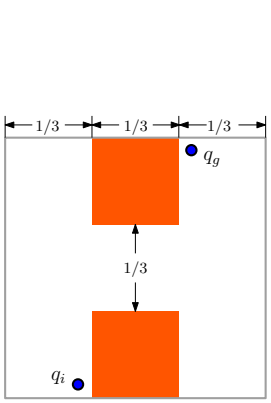
the milestones generated by different sampling sources but same sampling distribution lie in roughly the same parts of $\mathcal{F}$, while the milestones generated by different sampling distributions are spread differently. Hence, when the number of milestones is large, the difference in milestones placement generated by different sampling distributions is much more significant than that generated by different sampling source. As a result, as the narrow passage problem becomes more severe, the difference in the overall performance of sampling strategies with different sampling distribution is significantly more than that of sampling strategies with different sampling sources.

Similar results have been obtained in a more realistic example, *e.g.*, scene-2 where a six-dofs robot manipulator needs to access the bottom of a car through a narrow slot between the lift supports. These results indicate that sampling distribution plays the critical role in determining the overall efficiency of the planner.

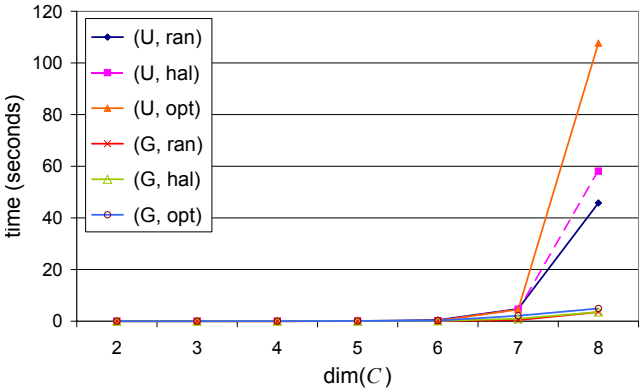
As the dimension of $\mathcal{C}$ increases

Another objective of improving probabilistic planning is to increase the number of dimensions $\dim(\mathcal{C})$ of $\mathcal{C}$ that can be handled by the planner well. For this, we ran two sets of tests. First, illustrated as scene-3 in Figure 2.6, we vary $\dim(\mathcal{C})$ directly. We generate $\mathcal{C}$ with two wide-open space separated by a passage. Each wide-open space occupies $1/3$ of the total volume of $\mathcal{C}$. While the passage occupied $(1/3)^{\dim(\mathcal{C})}$ of the total volume of $\mathcal{C}$. So, as $\dim(\mathcal{C})$ increases, the relative volume of the passage that connects the space in the left and right sides of the obstacles decreases. We vary the dimension from two to eight. The second test, illustrated as scene-4 in Figure 2.6, varies $\dim(\mathcal{C})$ by varying the number of links of a planar snake-like robot. In this scene, a planar snake-like robot has to move from a wide open space in the left to a wide-open space in the right, passing through a passage in-between the two obstacles. We increase the number of links of the robot from one to six, generating $\mathcal{C}$ with dimensions from three to eight.

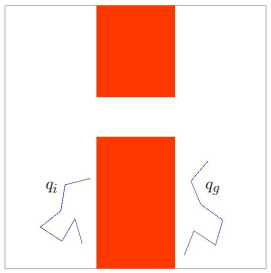
The results are shown beside the corresponding scenes in Figure 2.6. These results show that as $\dim(\mathcal{C})$ increases, the computation cost of all three Gaussian strategies increase slower than that of any of the strategies with uniform measure.



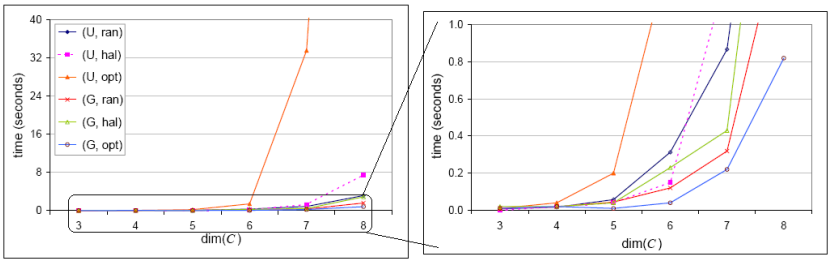
scene-3



results of scene-3



scene-4



results of scene-4

Figure 2.6: Scenarios and results of sampling distribution vs. sampling source as $\dim(\mathcal{C})$ increases.

This again indicates that sampling distribution is the more dominant factor in determining the overall efficiency of the planner.

The similar trends between the results in Figure 2.6 and the results for scene-1 in Figure 2.5 are not surprising, as both generate similar effect on the visibility property of $\mathcal{F}$. As in decreasing corridor width, increasing $\dim(\mathcal{C})$ worsen the visibility property of $\mathcal{F}$. The scenario in scene-3 shows that although the length of the passage in each dimension remains the same, *i.e.*, 1/3, increasing $\dim(\mathcal{C})$ reduces the relative volume of the passage. Which means, the relative size of the visibility set of the points inside the passage is reduced too. And hence, the visibility property of $\mathcal{F}$ worsen. Therefore, more milestones are needed to solve higher dimensional problems. And just as the case with our experiments with scene-1 and scene-2, the difference in the overall performance of sampling strategies with different sampling distribution is significantly more than that of

sampling strategies with different sampling sources.

Since sampling distribution turns out to have a more significant role on the overall performance of probabilistic path planning, for the rest of this thesis, we will focus on sampling distribution. Furthermore since quasi-random number does not show any significant advantage, the difficulty of generating various sampling distributions using quasi-random source discourages us from using quasi-random number generator as the sampling source Sr. So, for the rest of this thesis, we will use off-the-shelf pseudo-random number generator as a black-box for the sampling source.

2.5 The Desired Sampling Distribution

Despite the many works in narrow passage problem, the ideal sampling distribution for overcoming the problem remains obscure. However, the results of formal analyses [Kavraki et al., 1995, Kavraki et al., 1996a, Švestka, 1996, Hsu et al., 1997, Ladd and Kavraki, 2004, Chaudhuri and Koltun, 2007] on the performance of probabilistic path planning give us insights on what the desired sampling distribution is. Intuitively, the desired sampling distribution assigns high sampling density to regions of the free-space $\mathcal{F}$ with small visibility set and vice-versa. A configuration $q' \in \mathcal{F}$ is in the visibility set of another configuration $q \in \mathcal{F}$ whenever the straight line segment $\overline{qq'}$ between q and q' lies entirely in $\mathcal{F}$. Before discussing the desired sampling distribution further, we first present a brief survey on the formal analyses on the performance of probabilistic path planning.

Intuitively, the performance of probabilistic path planners depends critically on the quality of the roadmap $\mathcal{R}$ it generates. A good roadmap has two important properties, *i.e.*, *adequate coverage* and *correct connectivity*. Adequate coverage means that for any configuration $q \in \mathcal{F}$, there is a collision-free straight-line path between q and a milestone in $\mathcal{R}$ with high probability. Correct connectivity means that for any two milestones of $\mathcal{R}$ that lie in the same connected component of $\mathcal{F}$, they must also be connected by a path in $\mathcal{R}$. If $\mathcal{R}$ does not satisfy any of the two properties above, the planner will generate false negative for many queries.

Several works [Kavraki et al., 1995, Kavraki et al., 1996a, Švestka, 1996, Hsu et al., 1997, Ladd and Kavraki, 2004, Chaudhuri and Koltun, 2007] have tried to formally articulate the above intuition. In general, all of these articulations are based on some notion of the visibility property of the robot's free-space $\mathcal{F}$.

In [Kavraki et al., 1995] and [Švestka, 1996], the number of milestones needed by the basic-PRM to generate a roadmap that adequately covers $\mathcal{F}$ is defined in terms of the volume of the smallest visibility set of a point in $\mathcal{F}$. This property is referred to as the ϵ-goodness property. The ϵ-goodness notion is extended to expansiveness in [Hsu et al., 1997] to define the performance of the basic-PRM in both covering and capturing the connectivity of $\mathcal{F}$. It analyzes the performance in terms of the volume of the visibility set and the volume of the lookout region. Intuitively, a lookout region of a subset $G \subseteq \mathcal{F}'$ of a connected component $\mathcal{F}'$ of $\mathcal{F}$ is the region of $\mathcal{F}' \backslash G$ that is visible from the points in G. The connected component $\mathcal{F}'$ where the smallest visibility set and lookout regions are relatively large, compared to the volume of $\mathcal{F}'$, can be well represented by a roadmap with smaller number of milestones. We will call such regions as regions with favourable visibility property.

Instead of analysing the shape of $\mathcal{F}$, another analysis [Kavraki et al., 1996a] focuses on analyzing the property of the path between two configurations. It defines the probability that the basic-PRM finds a path between the given query in terms of the length of the path and the distance between the path and its nearest obstacle. This work has been extended to be applicable for analyzing probabilistic planner with any sampling distribution and not just uniform distribution in [Ladd and Kavraki, 2004]. Although it does not explicitly use the visibility property of $\mathcal{F}$, the distance between a point to its nearest obstacle is related to the volume of the visibility set of $\mathcal{F}$. If the distance from a point to its nearest obstacle is large, then the visibility set of this point must be large too.

Recently, smoothed analysis has been used to analyze the performance of the basic-PRM [Chaudhuri and Koltun, 2007]. Smoothed analysis computes the worst case over inputs of the expected running time of an algorithm under random perturbation of the inputs. Suppose $\mathcal{C}$ is an n-dimensional Euclidean space where

the forbidden regions are polyhedra bounded by k $(n-1)$-simplices whose vertices are perturbed by Gaussian distribution with variance σ^2. Then, the number of samples needed to construct an accurate roadmap is polynomial in k and $\frac{1}{\sigma}$. Since the smoothly perturbed free-space can be conjectured as an expansive space [Chaudhuri and Koltun, 2007], the results of this analysis is related to the visibility property of $\mathcal{F}$ too.

The above results indicate that narrow passages are regions of the robot's free-space $\mathcal{F}$ with poor visibility property. Intuitively, $\mathcal{F}$ has poor visibility property when there is a path-connected component $\mathcal{F}'$ in $\mathcal{F}$ where a large subset of $\mathcal{F}'$ is visibile from only a small region of $\mathcal{F}'$. These analyses show that more milestones are needed to generate a good roadmap when $\mathcal{F}$ has poor visibility property and vice-versa. To find a desired sampling distribution, one can utilize the above results by thinking of partitioning $\mathcal{F}$ into subsets and computing the number of milestones needed to generate a good roadmap in each subset. The normalized number of milestones computed for each subset indicates the desired sampling distribution for the particular subset of $\mathcal{F}$. To simplify, we can use a sufficient condition that if a subset $\mathcal{F}''$ has smaller visibility set, then more milestones are needed to generate a good roadmap in $\mathcal{F}''$. So intuitively, the *desired sampling distribution* assigns higher sampling density to subsets of $\mathcal{F}$ with smaller visibility set.

The above intuition of the desired sampling distribution is strengthen by several empirical studies [Geraerts and Overmars, 2002, Geraerts and Overmars, 2005, Morales et al., 2006] that have empirically compared the performance difference between various sampling strategies and by our own empirical study presented in Section 2.4. The results of those empirical studies suggest that in narrow passage problem, sampling strategies that vary their sampling distributions based on an estimated size of the visibility set of the robot's free-space tend to outperform those that do not.

In the next section, we present a short survey on the various sampling strategies that have been proposed to alleviate the narrow passage problem.

2.6 Sampling Strategies in Probabilistic Path Planning

In this section, we present a short survey on the various sampling strategies that have been proposed for probabilistic path planning. We clarify the distinction between sampling distribution and sampling source, which has been blurred in the probabilistic path planning literature. By doing so, we are able to classify previous sampling strategies into either modifying sampling distribution or sampling source.

A summary of the various sampling strategies for probabilistic path planning is shown in Figure 2.7.

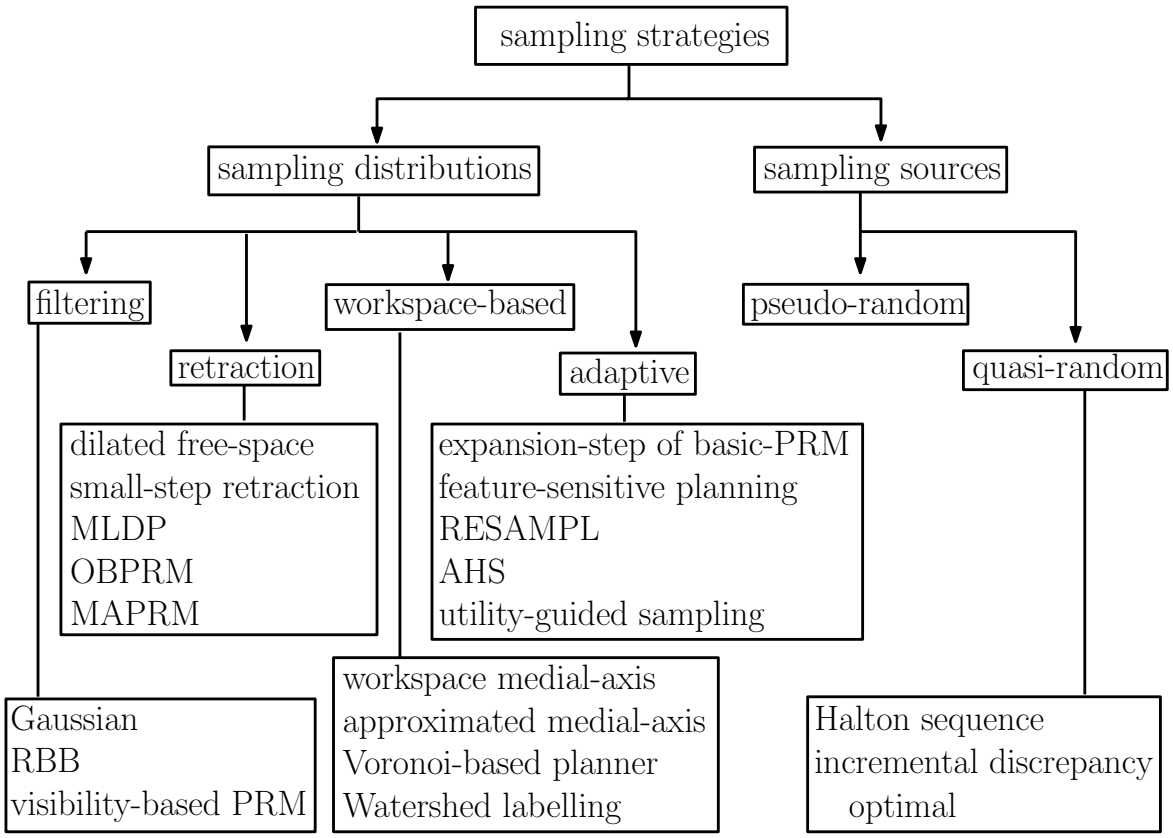


Figure 2.7: Various sampling strategies for probabilistic path planning.

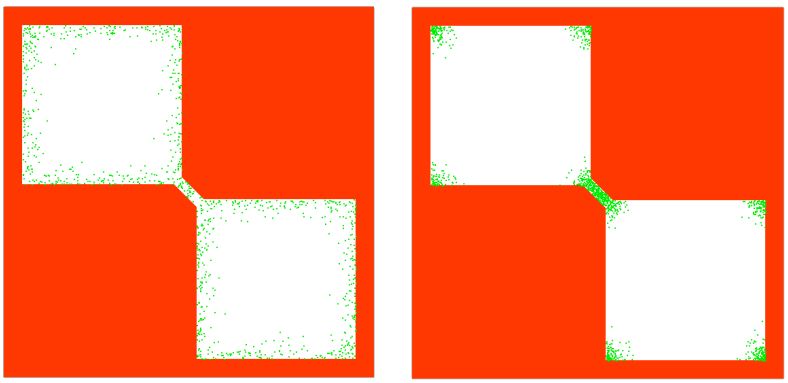
2.6.1 Sampling Strategies with Non-Uniform Measure

In this section, we review the various probabilistic path planners that improve the basic-PRM by modifying Pr. We classify them based on the information they use

and how they use these information.

Filtering strategies

Filtering strategies use a certain local geometric pattern of $\mathcal{F}$ to quickly reject many unpromising samples. The idea is based on an observation that in general, `FreePath`, *i.e.*, checking whether an edge between two milestones is collision-free, is the most costly operation in constructing a roadmap [Kavraki et al., 1996b]. By rejecting unpromising samples from becoming milestones in the roadmap, filtering strategies reduce the number of milestones, and hence reduce the number of calls to `FreePath`. These strategies include Gaussian strategy [Boor et al., 1999], randomized bridge builder (RBB) [Sun et al., 2005], and visibility-based PRM [Siméon et al., 2000].



(a) Gaussian strategy (b) RBB

Figure 2.8: Milestone placement of Gaussian strategy vs. RBB. 1000 sampled collision-free configurations (shown as the green dots) using Gaussian and RBB. The free-space is colored white.

Gaussian strategy [Boor et al., 1999] is based on two observations, *i.e.*, configurations with poor visibility often lie close to the boundary of $\mathcal{F}$ and `FreeConf` uses less computation time than `FreePath`. Utilizing these observations, Gaussian tries to locate the boundary of $\mathcal{F}$ using `FreeConf`, and samples more densely there. It samples a pair of configurations from $\mathcal{C}$. The first configuration q is sampled using uniform distribution over $\mathcal{C}$. The second configuration is sampled such that its distance to q forms a normal distribution with zero mean and constant standard

deviation. Next, Gaussian applies `FreeConf` to check each of the two configurations. If exactly one of the two configurations is collision-free, the collision-free configuration is kept as a milestone in the roadmap. Otherwise, both configurations are discarded. In summary, Gaussian strategy uses a few calls to the cheaper primitive `FreeConf` to place milestones at a more strategic location, such that the number of milestones and hence the number of calls to `FreePath` are reduced.

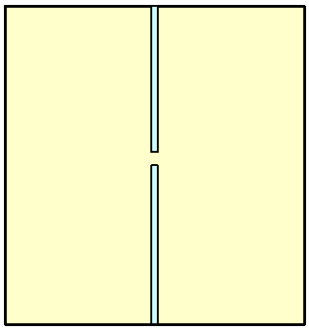


Figure 2.9: An illustration of $\mathcal{C}$ where Gaussian and RBB perform poorly. The free-space is colored light yellow.

RBB [Sun et al., 2005] uses the same idea of trading calls to `FreePath` with a few calls to `FreeConf`, but it uses a different geometric pattern. It is based on an observation that many regions that are close to the boundary of $\mathcal{F}$ have good visibility property, and those that really have poor visibility property lie in-between two or more forbidden regions. To use this geometric pattern, RBB starts by sampling a pair of configurations q and q' just as Gaussian strategy. RBB will also check both q and q' for collision. But unlike Gaussian, RBB looks for a pair of in-collision configurations. If both q and q' are in-collision, the mid-point will be kept as a milestone in the roadmap whenever it is collision-free. An illustration of the difference between Gaussian and RBB is shown in Figure 2.8.

Although the two strategies above are simple to implement, they have difficulties in sampling collision-free configurations when the forbidden regions are thin, because sampling a configuration from the forbidden region is already difficult. An example is shown in Figure 2.9.

The Visibility-based PRM [Siméon et al., 2000] uses a slightly different idea. It does not trade calls to `FreePath` with calls to `FreeConf`. Instead, it uses `FreePath` to estimate the visibility set of a configuration and only keeps samples that improve the coverage or connectivity of the current roadmap as milestones. By doing so, it reduces the number of milestones in regions that are already well represented by the current roadmap, and hence reduces the number of calls to `FreePath` in the future. The main problem with this method is that the cost for filtering is quite

expensive such that filtering may not show significant benefit.

Retraction strategies

Retraction strategies "pull" sampled configurations toward a more desirable regions of $\mathcal{F}$. The motivation is similar to that of snowball sampling [StatPac, 1997] in survey sampling, where reference from previous samples are used to help sample the more difficult region. Most methods in this strategy, *e.g.*, Obstacle-Based PRM (OBPRM) [Amato et al., 1998], dilated free-space [Hsu et al., 1998], small-step retraction [Saha and Latombe, 2005], and Multi-Level Dilation Planner (MLDP) [Cheng et al., 2006] retract in-collision samples toward $\mathcal{F}$. Only Medial Axis PRM (MAPRM) [Wilmarth et al., 1999] retracts all samples toward the approximated medial axis of $\mathcal{F}$.

OBPRM [Amato et al., 1998] retracts all in-collision configurations. It starts by sampling $\mathcal{C}$ using uniform distribution. Then, for each in-collision configuration q, OBPRM picks a direction θ emanating from q uniformly at random. It then finds a collision-free configuration along θ, that lies near the boundary of $\mathcal{F}$.

Instead of retracting all in-collision configurations, Dilated free-space [Hsu et al., 1998], small-step retraction [Saha and Latombe, 2005], and MLDP [Cheng et al., 2006] retract in-collision configurations that lie within a pre-defined distance from the boundary of $\mathcal{F}$. Dilated free-space decides whether an in-collision configuration q should be retracted, by computing the penetration depth between the robot (placed at q) and obstacles in the workspace. Only in-collision configurations with penetration depth within a fixed constant are retracted. The retraction is performed by sampling a region of fixed size and geometry around q, uniformly at random. On the other hand, small-step retraction and MLDP pre-compute one or more shrunken versions of the robot. Only in-collision configurations that lie in the free-space of the shrunken robot are retracted. The retraction is then performed as in the dilated free-space strategy.

The idea behind the above three retraction strategies is based on the fact that uniform sampling performs much better when $\mathcal{F}$ has good visibility property. By dilating $\mathcal{F}$, these strategies enlarge the visibility set of the regions of $\mathcal{F}$. Hence,

the dilated free-space has a more favorable visibility property. These strategies use uniform sampling to find a path in the dilated free-space, and then deform the path such that it is collision-free in the original free-space.

MAPRM [Wilmarth et al., 1999] retracts all sampled configurations toward the approximated medial axis of the free-space $\mathcal{F}$. It starts by sampling $\mathcal{C}$ using uniform distribution. Then, for each configuration q, MAPRM finds a point q' that lies on the boundary of $\mathcal{F}$ and is nearest to q. It then retracts q to the first collision-free configuration on the line segment $\overline{qq'}$ that has more than one nearest point on the boundary of $\mathcal{F}$.

The rationale behind MAPRM is that points near the medial axis have large clearance, and hence in general have better visibility property. Furthermore, medial axis of $\mathcal{F}$ captures the connectivity of $\mathcal{F}$. Hence, one may expect that a roadmap where the milestones lie near to the medial axis of $\mathcal{F}$ captures the correct connectivity of $\mathcal{F}$ with a relatively small number of milestones, and hence with less cost.

The main drawback of retraction strategies is that they often require complicated and costly computation either to decide whether to retract in-collision configurations or to perform the retraction.

Workspace-based strategies

As the name implies, workspace-based strategies use information from workspace to bias sampling in $\mathcal{C}$. The idea is based on two observations. First, in general, forbidden regions are caused by obstacles in the workspace. Based on this, one can expect that the geometry and connectivity of the workspace free-space $\mathcal{W}_{\mathcal{F}}$, *i.e.*, workspace regions that are not occupied by obstacles, may give some hints on the geometry and connectivity of $\mathcal{F}$. The second observation is that exact workspace representation is explicitly available and has low dimension, *i.e.*, three at most. Explicit representation and low dimensionality enable one to extract the necessary geometric and/or topological information of $\mathcal{W}_{\mathcal{F}}$, efficiently. Thus, the main idea of workspace-based strategies is to use efficient computation to extract geometric and/or topological properties from $\mathcal{W}_{\mathcal{F}}$, and then use this information to infer

the visibility property of $\mathcal{F}$. These strategies include Workspace Medial Axis for PRM [Holleman and Kavraki, 2000], Approximated Medial Axis method [Yang and Brock, 2004], Voronoi-based planner [Foskey et al., 2001], and Watershed labeling algorithm [van den Berg and Overmars, 2005].

Both Workspace Medial Axis for PRM [Holleman and Kavraki, 2000] and Approximated Medial Axis method [Yang and Brock, 2004] retract samples in $\mathcal{C}$ to be near to the medial axis of $\mathcal{W}_{\mathcal{F}}$. They start by sampling configurations from $\mathcal{C}$ using uniform distribution and then retract these configurations such that several points on the robot lie near the medial axis of $\mathcal{W}_{\mathcal{F}}$.

Voronoi-based planner [Foskey et al., 2001] is designed for a rigid-body robot that can both translate and rotate. It finds a path in the workspace for a point a on the robot, and then uses this workspace path to find a collision-free path for the whole robot. It starts by constructing a Voronoi graph of $\mathcal{W}_{\mathcal{F}}$ and finding a workspace path for a in this graph. After a path in the workspace is found, for each Voronoi vertex w in the workspace path, the planner uses finite difference estimate to find a rotation such that the configuration places point a of the robot at w and is collision-free. When a corresponding collision-free configuration cannot be found or when the configurations in $\mathcal{F}$ cannot be connected with a collision-free straight line segment, the planner performs a refinement step by sampling around the invalid part of the path. If this step still results in failure, the planner samples from $\mathcal{C}$ using EST [Hsu et al., 1999] and then uniform distribution as the last resort.

Watershed-labelling algorithm [van den Berg and Overmars, 2005] uses workspace information to indicate small regions with large lookout in $\mathcal{W}_{\mathcal{F}}$, and then sample more densely there. It assumes that small regions with large lookout in $\mathcal{F}$ are generated by small passages that connects two or more wide-open space in $\mathcal{W}_{\mathcal{F}}$. This strategy decomposes $\mathcal{W}_{\mathcal{F}}$ into cells and assigns higher weight whenever a small cell is adjacent to two or more large cells. It then samples regions of $\mathcal{C}$ according to the weights assigned to its corresponding regions in the workspace.

All of the proposed workspace-based strategies a priori determine a sampling distribution based on information from the workspace alone. This is often in-

adequate because although the workspace may give hints on the geometry and connectivity of $\mathcal{F}$, the hints may be misleading. Afterall, $\mathcal{F}$ is generated from the interaction between the robot and the workspace, and not the workspace alone. When the information from the workspace is misleading, the generated sampling distribution is unlikely to be suitable for the problem at hand.

Adaptive strategies

Adaptive strategies adapt its sampling distribution according to the information gathered during roadmap construction. These strategies exploit the observation that in probabilistic planning, the goal of sampling is to gather witnesses [Motwani and Raghavan, 2000], *i.e.*, to find a set of configurations that represents $\mathcal{F}$ well, and not to approximate some value(s). Which means, keep on sampling parts of $\mathcal{F}$ that have been well represented by the current roadmap is not useful. Adaptive strategies include the expansion step of the basic-PRM planner [Kavraki et al., 1996b], feature-sensitive planning [Morales et al., 2004], region-sensitive adaptive motion planner (RESAMPL) [Rodriguez et al., 2006], adaptive hybrid sampling (AHS) [Hsu et al., 2005], and utility-guided sampling [Burns and Brock, 2005].

The expansion step in the basic-PRM [Kavraki et al., 1996b] resamples $\mathcal{C}$ around existing milestones that is predicted to lie in region with poor visibility. When the basic-PRM samples configurations uniformly over $\mathcal{C}$ to construct the roadmap, for each milestone, it keeps a failure ratio of connection. The failure ratio for a milestone m is computed as the number of calls to `FreePath`(m, q) or `FreePath`(q, m) that returns false over the total number of such calls, where q is other milestones in the roadmap. A weight proportional to the failure ratio is then assigned to each milestone. This means, milestones in regions of poor visibility will get higher weight. During expansion step, basic-PRM selects milestones to be expanded, randomly according to its weight. Suppose m is one of the milestones selected. The basic-PRM will then sample $\mathcal{C}$ around m using random walk starting from m. By doing so, the basic-PRM increases sampling distribution over regions of $\mathcal{F}$ that have poor visibility property, in order to improve connectivity of the roadmap.

Feature-sensitive planning [Morales et al., 2004] and RESAMPL [Rodriguez et al., 2006] propose a more sophisticated expansion strategy. As the basic-PRM, these two strategies start by sampling uniformly over $\mathcal{C}$. However, they keep much more statistics than only failure ratio of connection, *e.g.*, free-node ratio, number of components, edge length, etc. These statistics are used to divide $\mathcal{C}$ into regions and classify these regions. The planner will then assign a suitable sampler to each region. And sampling is continued by selecting a region to be sampled and applying the assigned sampler to sample a free configuration from the region.

AHS [Hsu et al., 2005] also uses multiple sampler to sample $\mathcal{C}$. However, it combines these samplers using a reinforcement learning strategy. To sample a configuration, AHS starts by selecting which sampler to use, randomly according to the weight assigned to each sampler. The weight of a sampler is based on the rewards it gathers. A positive reward is given to a sampler every time it samples a configuration that improves the coverage and/or connectivity of the current roadmap. As roadmap construction progresses, some regions of $\mathcal{F}$ become well-represented. Samplers that perform well in sampling these regions will stop getting positive reward and hence their weight decreases. So, the sampler favored by AHS changes as the roadmap progresses.

A slightly different approach is used in utility-guided sampling [Burns and Brock, 2005]. It incrementally constructs an approximate model of $\mathcal{C}$ using a collection of Gaussian distributions. The model is constructed based on the sampled configuration and is used to guide future sampling. Utility-guided sampling samples configurations with the highest potential for improving the current approximated model.

All of the proposed adaptive strategies are based on sampling history alone. Since a sampled configuration q can only inform us about the local geometry of $\mathcal{C}$ surrounding q, to learn the usefulness of sampling a particular region of $\mathcal{C}$, we need many samples from or around that region. This is difficult to achieve in narrow passages. Hence, to identify and favor sampling from a narrow passage region, the above strategies may require a long learning time.

A summary of the various sampling strategies is presented as a diagram in

Figure 2.7.

2.6.2 Sampling Source

To sample a configuration in $\mathcal{C}$ according to a probability measure Pr, a probabilistic planner uses the sampling source Sr to generate a point in a unit hypercube of suitable dimensions, and then maps the point to a configuration in $\mathcal{C}$ according to Pr. Since most probabilistic planners use random sampling, they use random number generator as Sr. A random number generator generates a sequence of numbers that are uniformly distributed in the real line and satisfy several statistical properties of randomness [Knuth, 1998]. The random numbers generated are often called pseudo-random numbers as they only "look" random, based on statistical testing, but they are generated by a deterministic sequence of operations in a computer. Many probabilistic planners use off-the-shelf random number generator as Sr and treat it as a black-box.

However, by clarifying the distinction between sampling distribution and sampling source, it becomes clear that recent work [LaValle et al., 2004], that has started to become a trend in probabilistic path planning, has argued that carefully tailoring Sr for path planning purposes will significantly improve probabilistic planning. The main argument is that by carefully tailoring Sr, one can generate a resolution complete planner (*i.e.*, if a problem can be solved at a particular resolution, at that resolution, the planner will solve the problem) and in general this planner is faster than planners with pseudo-random number generator. The work in [LaValle et al., 2004] proposes to replace pseudo-random number generator with quasi-random number generator that minimizes discrepancy and dispersion of the generated numbers.

Discrepancy and dispersion measure how evenly spread a set of points P distributed in a Euclidean space E are [Niederreiter, 1992]. Discrepancy is based on volume and is defined as

$$D(P, \mathcal{V}) = \sup_{V \in \mathcal{V}} \left| \frac{|P \cap V|}{|P|} - \mu(V) \right|$$

where $\mathcal{V}$ is a collection of subsets in E, $|\cdot|$ denotes the number of points, and $\mu(\cdot)$ denotes volume. Intuitively, discrepancy measures how well a set of points in a space is, in approximating uniform distribution over the space. Dispersion, on the other hand, is based on distance metric and is defined as

$$\rho(P, \delta) = \sup_{e \in E} \min_{p \in P} \delta(e, p)$$

where δ is any metric in E. Intuitively, dispersion measures the radius of the largest empty ball, *i.e.*, ball without any points from P inside, in the space. Low discrepancy implies low dispersion.

Two of the quasi-random number that have been shown to perform well for probabilistic path planning [Lindemann and LaValle, 2003, LaValle et al., 2004] are Halton sequence [Niederreiter, 1992] and incremental discrepancy-optimal [Lindemann and LaValle, 2003]. Halton sequence has low discrepancy and hence low dispersion, too. In 1-dimension, a Halton sequence is generated by counting in base-2 and then reversing the least significant bit. For example, to generate the first four numbers of Halton sequence in $[0, 1]$, one starts by counting in base-2 to generate a sequence of 0.00, 0.01, 0.10, and 0.11. Next, the least significant bits are reversed to yield 0.00, 0.10, 0.01, and 0.11. So, the generated first four numbers of Halton sequence in $[0, 1]$ is 0, 0.5, 0.25, and 0.75. For n-dimensions, Halton sequence performs the same procedure to each dimension, but different dimension uses different prime-number as the base. Incremental discrepancy-optimal divides the space into grids according to a suitable resolution and places points at the lower-left corner of each grid. However, the order on which points appear first is based on a special ordering [Lindemann and LaValle, 2003] to maintain low discrepancy throughout the whole sampling process.

2.7 Where to go next?

In the previous sections, we have discussed the importance of sampling distribution in probabilistic path planning. We have clarified the distinction between sampling distribution and sampling source, which has been blurred in probabilistic

path planning literature. And we have presented empirical study that shows the importance of a suitable sampling distribution in alleviating the narrow passage problem. We have also discussed that the desired sampling distribution depends on the visibility property of the robot's free-space. One question remains as to how do we construct a suitable sampling distribution, considering the shape of $\mathcal{F}$ and hence its visibility property are unknown.

Since the shape of $\mathcal{F}$ is unknown, the most common approach to approximate the desired sampling distribution is to a priori guess the location of subsets of $\mathcal{F}$ with unfavorable visibility property and then sample more densely there. In fact, almost all of the non-uniform sampling strategies presented in Section 2.6.1 improve the basic-PRM by doing so. Nevertheless, this is still an open problem.

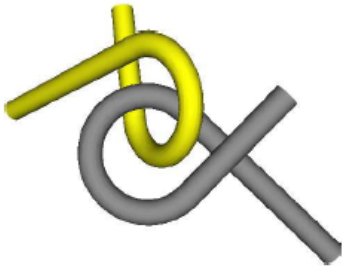


Figure 2.10: The alpha-puzzle consists of two identical rigid tubes. One act as the robot while the other as the obstacle. The geometry of the two tubes are intentionally designed to create a non-obvious narrow passage between the configuration when the tubes are intertwined and when they are separated.

Although the exact free-space is unknown, forbidden region in the configuration space can be considered as a convolution between the robot and obstacles in the workspace. So, it is reasonable to assume that workspace provides a rough information on the shape of the free-space and hence provides a rough indication of the location of narrow passages too. Furthermore, explicit representation and the low dimensionality of the workspace allow us to efficiently extract its geometric property. Therefore, it is reasonable to expect that we can use efficient geometric computation to extract geometric property of the workspace and then use this information to roughly identify the location of narrow passages in $\mathcal{F}$.

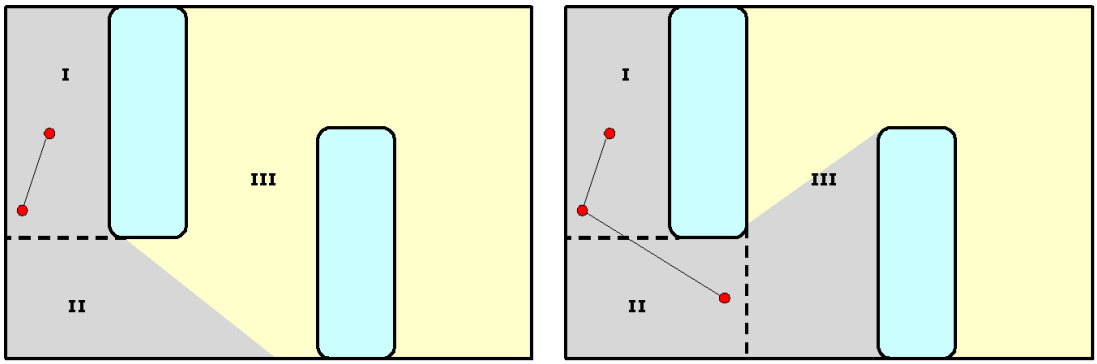
It is also interesting to note that recent result of smoothed analysis of probabilistic path planning [Chaudhuri and Koltun, 2007] suggests that narrow passages in the high-dimensional free-space are often caused by narrow passages that can be seen in the workspace. Intuitively, a narrow passage in the high-dimensional

free-space occurs when the boundaries of the forbidden region almost coincide. Hence, small random perturbation on the boundary of the forbidden region, which can occur due to small random perturbation of the workspace geometry, is enough to enlarge or even eliminate the narrow passage. This indicates that generating a non-obvious narrow passage such as the one in Figure 2.10 is not easy. Instead, in many real world examples, narrow passages in $\mathcal{F}$ occurs because of the intentional narrow passages generated in the workspace, *e.g.*, when a robot needs to pass through a small opening or when two mechanical parts must be assembled together by inserting one into another. This recent result strengthen our intuition that workspace provides useful information for generating suitable sampling distributions that alleviate narrow passage problem in probabilistic path planning.

Despite the above potentials and although several sampling strategies have utilized workspace information to generate sampling distribution (see Section 2.6.1), the use of this information has not been explored much. For instance, although distance between obstacles is the most obvious indication of the existence of narrow passages in the workspace, no sampling strategy have exploited this information. The above considerations encourage us to explore further on the use of workspace information in generating suitable sampling distribution for probabilistic path planning.

Another strategy of improving probabilistic planning is to dynamically adapt the sampling distribution. As discussed in Section 2.3, the main objective of probabilistic planning is to converge quickly to the set of consistent hypotheses that has the largest set of solvable queries. And the sampling distribution represents the planner's belief on the advantage of sampling a particular subset of $\mathcal{F}$ in enlarging the set of solvable queries. Simple observation indicates that the advantage of sampling a subset of $\mathcal{F}$ may change over time, depending on the current roadmap. Figure 2.11 shows an illustration. This observation encourages us to push forward the idea of generating dynamic sampling distribution instead of static sampling distribution.

As a summary, the results in this chapter suggest that workspace information and dynamically adapting the sampling distribution over time are two promising



(a) (b)

Figure 2.11: Illustration on expansion of the set of solvable queries. The free-space $\mathcal{F}$ is colored light yellow, $\mathcal{C}\backslash\mathcal{F}$ is colored light green. The milestones are the red dots and the visibility set of the roadmap are colored light grey. (a) In this state of the roadmap, sampling from the region marked as I will not enlarge the set of solvable queries by much. However, sampling from II or III may still enlarge the set of solvable queries, significantly. (b) As more milestones are added to the roadmap, sampling from II will no longer enlarge the set of solvable queries significantly. And it would be more beneficial to sample from III than from I or II.

avenues for improving current probabilistic path planning in handing narrow passage problem. In the next chapter, we explore the use of workspace information in generating suitable sampling distribution for probabilistic path planning. We explore the relation between the visibility sets of point in the free-space and in the workspace. And we present a simple strategy for utilizing workspace information called WIS. Then in Chapter 4, we present a new probabilistic path planner called WCO that uses workspace information to generate *dynamic* sampling distribution.

Chapter 3

Exploring Workspace Information

The main purpose of this chapter is to explore the use of workspace information in generating suitable sampling distribution over the high-dimensional configuration space. Since the suitability of a sampling distribution depends on the visibility property of the robot's free-space $\mathcal{F}$ (Chapter 2), we start by exploring the relation between the visibility property of the workspace free-space $\mathcal{W}_{\mathcal{F}}$ and the free-space $\mathcal{F}$. We show that under certain conditions that *do not* depend on the dimensionality of $\mathcal{F}$, the visibility sets of points in $\mathcal{W}_{\mathcal{F}}$ are highly related to those of points in $\mathcal{F}$. We then use the relation between the visibility sets of points in $\mathcal{W}_{\mathcal{F}}$ and $\mathcal{F}$ to construct a simple workspace-based probabilistic path planner, called Workspace Importance Sampling (WIS). WIS uses local geometric property of $\mathcal{W}_{\mathcal{F}}$ to estimate the size of the visibility sets of points in $\mathcal{W}_{\mathcal{F}}$ and uses this estimation to generate a static sampling distribution over $\mathcal{C}$. Our experimental results show that WIS is competitive with the recent probabilistic path planner [Hsu et al., 2005] that has been shown to perform well in solving narrow passage problem. Our analysis shows that the failure probability of WIS converges to zero exponentially in the number of milestones, whenever a solution exists. Furthermore, in general when the path requires the robot to move inside the narrow passages of $\mathcal{W}_{\mathcal{F}}$, the upper bound of the failure probability of WIS is lower compared to that of the basic-PRM [Kavraki et al., 1996b]. In short, the results in this chapter encourage us to exploit workspace information further.

In Section 3.1 and Section 3.2, we present the notations needed to relate the

configurations in $\mathcal{F}$ with the points in $\mathcal{W}_\mathcal{F}$. Next, we present the relation between the visibility sets of points in $\mathcal{F}$ and in $\mathcal{W}_\mathcal{F}$ in Section 3.3. Then, we present our simple workspace-based probabilistic planner WIS in Section 3.4. Implementation details and experiments along with analysis of WIS are then presented in Section 3.5 and Section 3.6, respectively. Finally in Section 3.7, we end this chapter by a discussion on how to exploit workspace information further to generate better probabilistic path planner.

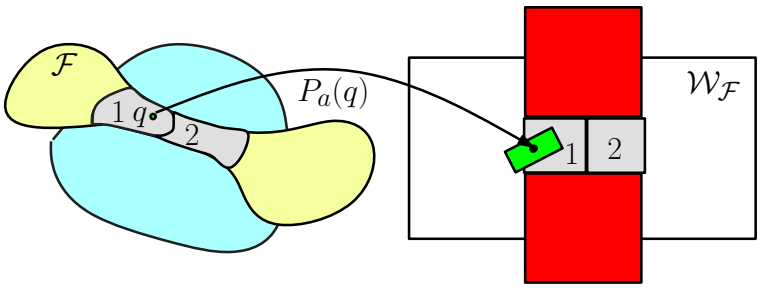
3.1 Notations for Connecting $\mathcal{F}$ and $\mathcal{W}_\mathcal{F}$

Before we can show the relations between $\mathcal{F}$ and $\mathcal{W}_\mathcal{F}$, we need some notation to articulate the mapping between points in $\mathcal{C}$ and points in $\mathcal{W}$. For a point a in a robot $\mathcal{A}$, let $P_a(q)$ be the position of point a in $\mathcal{W}$ when $\mathcal{A}$ is placed at configuration $q \in \mathcal{C}$. We call the mapping $P_a{:}\mathcal{C} \rightarrow \mathcal{W}$ a projection, as the dimension of $\mathcal{C}$ is more than or equal to the dimension of $\mathcal{W}$. Similarly, we define the lift mapping $L_a{:}\mathcal{W} \rightarrow 2^\mathcal{C}$. For any $\mathbf{x} \in \mathcal{W}$, $L_a(\mathbf{x})$ is the subset of $\mathcal{C}$ such that each configuration in $L_a(\mathbf{x})$ places a at $\mathbf{x}$. For convenience, we extend the definitions of P_a and L_a to subsets of $\mathcal{C}$ and $\mathcal{W}$, respectively, by taking set union. An illustration is shown in Figure 3.1.

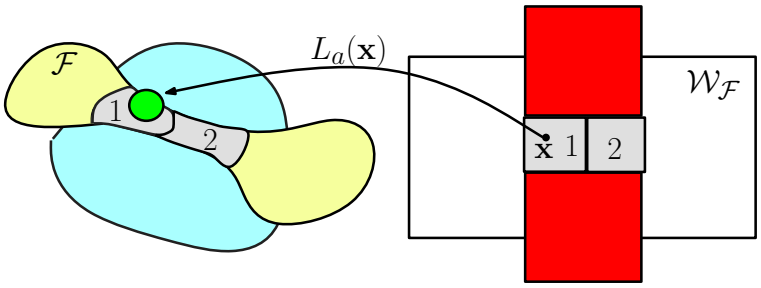
Notice that a partition in $\mathcal{W}_\mathcal{F}$ induces a partition in $\mathcal{F}$. Suppose $\mathcal{W}_\mathcal{F}$ is partitioned into cells and the set of all these cells is denoted as $\mathcal{T}_\mathcal{F}$. Then, for a fixed point a on the robot, $\mathcal{T}_\mathcal{F}$ induces a partition of the collision-free subset of $\mathcal{C}$ into equivalent classes: $\mathcal{F} = \bigcup_{t \in \mathcal{T}_\mathcal{F}} (L_a(t) \cap \mathcal{F})$ and for all $t, t' \in \mathcal{T}_\mathcal{F}$ and $t \neq t'$, $L_a(t) \cap L_a(t') = \emptyset$ unless when t and t' share a boundary, in which case $L_a(t)$ and $L_a(t')$ share a boundary too (Figure 3.1 shows an illustration). Two configurations are in the same equivalent class if they project to the same cell in $\mathcal{T}_\mathcal{F}$.

3.2 Distance in $\mathcal{C}$ and in $\mathcal{W}$

To relate the distance between configurations in $\mathcal{C}$ and the distance between points in $\mathcal{W}$, we introduce the distance function as follows. Euclidean distance is used as the distance function $\delta(\mathbf{x}, \mathbf{x'})$ between two points $\mathbf{x}$ and $\mathbf{x'}$ in $\mathcal{W}$. Furthermore,



(a)



(b)

Figure 3.1: Illustration of projection and lift mapping. The robot is the green rectangle, and point a is the black dot in the centroid of the robot. Any configuration in the region marked as 1 in $\mathcal{F}$ places a in the rectangle marked as 1 in $\mathcal{W}_\mathcal{F}$. We say that the region marked as 1 in $\mathcal{F}$ corresponds to the rectangle marked as 1 in $\mathcal{W}_\mathcal{F}$ and all configurations in the region marked as 1 in $\mathcal{F}$ belong to the same equivalent class. The same goes with the region marked as 2 in $\mathcal{F}$ and the rectangle marked as 2 in $\mathcal{W}_\mathcal{F}$. So, a partition of $\mathcal{W}_\mathcal{F}$ induces a partition of $\mathcal{F}$. In general, a connected region of $\mathcal{W}_\mathcal{F}$ may correspond to several disconnected regions of $\mathcal{F}$. (a) Illustration of $P_a(q)$. (b) Illustration of $L_a(\mathbf{x})$. Note that although $\mathbf{x}$ is in $\mathcal{W}_\mathcal{F}$, $L_a(\mathbf{x})$ may intersect the forbidden regions of $\mathcal{C}$.

we define the distance $\delta_\mathcal{O}(\mathbf{x})$ between a point $\mathbf{x}$ in $\mathcal{W}_\mathcal{F}$ and the obstacles in $\mathcal{W}$ as the shortest distance between $\mathbf{x}$ and its nearest point on the obstacles, *i.e.*, $\delta_\mathcal{O}(\mathbf{x}) = \min_{\mathbf{o}\in\mathcal{O}} \delta(\mathbf{x}, \mathbf{o})$, where $\mathcal{O}$ denotes the set of all obstacles in $\mathcal{W}$.

In $\mathcal{C}$, we define the distance $\Delta(q, q')$ between two configurations q and q' as the maximum distance traversed in $\mathcal{W}$ by a point on the robot when following the straight line segment between q and q' [Hsu et al., 1999]. Note that this definition of distance in $\mathcal{C}$ is only used for relaxing several theorems. In general, we use Euclidean distance for both $\mathcal{W}$ and $\mathcal{C}$, unless otherwise stated. Furthermore, we define the distance between a configuration $q \in \mathcal{F}$ and the forbidden region as

$\Delta_{\mathcal{O}}(q) = \min_{a \in \mathcal{A}} \delta_{\mathcal{O}}(P_a(q))$.

3.3 Visibility Properties of $\mathcal{F}$ and $\mathcal{W}_{\mathcal{F}}$

In this section, we show the relation between the visibility sets of points in $\mathcal{F}$ and those in $\mathcal{W}_{\mathcal{F}}$. In particular, we show that,

1. For any robot and for any point a on the robot such that the projection P_a is a linear transformation, the visibility set $\mathcal{V}(q)$ of a configuration $q \in \mathcal{F}$ is a subset of the lifted visibility set of the projection $P_a(q)$ of q in $\mathcal{W}_{\mathcal{F}}$, *i.e.*, $\mathcal{V}(q) \subseteq L_a(\mathcal{V}(P_a(q)))$. This result implies that to find a collision-free configuration that can be connected with a straight line segment to q, reducing our search space to the set of configurations that places point a of the robot at $\mathcal{V}(P_a(q))$ will not cause us to loose any possible solution.

2. For any robot and for any point a on the robot such that the projection P_a is defined as $P_a(q) = V_a \cdot q + \mathbf{b}$, where V_a is a $(d \times n)$-matrix ($d = \dim(\mathcal{W})$, $n = \dim(\mathcal{C})$), $rank(V_a) = d$, and $\mathbf{b}$ is a vector of d elements, the volume $\mu(\mathcal{V}(q))$ of the visibility set of a configuration $q \in \mathcal{F}$ is bounded by a constant multiplication of the volume of the visibility set of $P_a(q)$. Considering that the number of milestones for solving a path planning problem within a certain probability of success can be computed in terms of the volume of the visibility set of configurations in $\mathcal{F}$, our new result indicates that we can also bound the number of milestones in terms of the volume of the visibility set of points in $\mathcal{W}_{\mathcal{F}}$.

Considering that a suitable sampling distribution varies its sampling density based on the size of the visibility sets of configurations in $\mathcal{F}$ (Chapter 2), the above relations strengthen our intuition that workspace information is potentially useful for estimating a suitable sampling distribution over $\mathcal{C}$.

We start by showing the relation between the visibility set in $\mathcal{F}$ and in $\mathcal{W}_{\mathcal{F}}$ in Section 3.3.1. We then show the relation between the volume of the visibility set in $\mathcal{F}$ and in $\mathcal{W}_{\mathcal{F}}$ in Section 3.3.2. Next in Section 3.3.3, we elaborate on when the conditions for the above two relations are satisfied.

3.3.1 Visibility Sets of Points in $\mathcal{F}$ and in $\mathcal{W}_\mathcal{F}$

Now, we formally state the relation between the visibility sets of points in $\mathcal{F}$ and in $\mathcal{W}_\mathcal{F}$ as follows,

Theorem 3.1 *Let's denote $\mathcal{V}(\cdot)$ as the visibility set. Then, for any collision-free configuration q and for any point a in a robot $\mathcal{A}$ such that the mapping P_a is linear, $\mathcal{V}(q) \subseteq L_a(\mathcal{V}(P_a(q)))$.*

Proof By definition of the visibility set (see Section 2.5), a configuration $q' \in \mathcal{V}(q)$ means that $q' = q + r \cdot (q' - q)$ for $r \in [0, 1]$. Since a collision-free configuration q means that when the robot is at q, each point in the robot lies in the workspace free-space, then $P_a(q)$ and $P_a(q')$ must lie in the workspace free-space. And when P_a is linear, $P_a(q') = P_a(q) + r \cdot (P_a(q') - P_a(q))$. Thus, $P_a(q')$ is in the visibility set of $P_a(q)$. Taking the lift mapping gives the result we want. $\square$.

Intuitively, the above theorem means that given any robot, by choosing appropriate point on the robot such that the projection P_a is a linear mapping, the visibility sets of points in $\mathcal{F}$ and in $\mathcal{W}_\mathcal{F}$ are highly related. The visibility sets $\mathcal{V}(q)$ of a configuration $q \in \mathcal{F}$ must lie inside the lift-mapping of the visibility set $\mathcal{V}(P_a(q))$ of its projection $P_a(q)$ in $\mathcal{W}_\mathcal{F}$. This means that to find a collision-free configuration that can be connected with a straight line segment to q, we can safely restrict our search space based on the visibility set of $P_a(q)$ in $\mathcal{W}_\mathcal{F}$.

3.3.2 Volume of Visibility Sets of Points in $\mathcal{F}$ and in $\mathcal{W}_\mathcal{F}$

To show the relation between the volume of the visibility sets of points in $\mathcal{W}_\mathcal{F}$ and that of the visibility sets of points in $\mathcal{F}$, we first relate the volume of a subset $X \subseteq \mathcal{W}$ and the volume of its corresponding lift-mapping in $\mathcal{C}$. Without loss of generality, we represent $\mathcal{W}$ and $\mathcal{C}$ as normalized Euclidean space $[0, 1]^{\dim(\mathcal{W})}$ and $[0, 1]^{\dim(\mathcal{C})}$. The relation can then be stated more formally as,

Lemma 3.1 *Let's denote n as the dimension of $\mathcal{C}$, d as the dimension of $\mathcal{W}$, and $\mu(\cdot)$ as volume. Suppose a is a point on the robot such that P_a is a linear transformation, defined as $P_a(q) = V_a \cdot q + \boldsymbol{b}$, where V is a $(d \times n)$-matrix with*

$rank(V_a) = d$ and $\boldsymbol{b}$ is a vector of d elements. Then, for any subset $X \subseteq \mathcal{W}$, $\mu(L_a(X)) \leq K_a \mu(X)$ where K_a is a positive constant value that depends on a.

Proof The lift-mapping $L_a(\mathbf{x})$ of a point $\mathbf{x} \in X$ is the solution of

$$V_a \cdot q = \mathbf{x} - \mathbf{b} \tag{3.1}$$

Since $rank(V_a) = d$, the solution of (3.1) can be written in terms of $\mathbf{x}$ and $(n-d)$ independent parameters $p_1, \ldots, p_{n-d}$ as,

$$\begin{aligned} q &= \mathbf{u}_1 \cdot \mathbf{x}_1 + \ldots + \mathbf{u}_d \cdot \mathbf{x}_d + \mathbf{u}_{d+1} \cdot p_1 + \ldots + \mathbf{u}_n \cdot p_{n-d} + q_F \\ &= U_a \, \mathbf{p} + q_F \qquad ; \qquad U_a = [\, \mathbf{u}_1 \ \ldots \ \mathbf{u}_n \,] \ , \ \mathbf{p} = \begin{pmatrix} \mathbf{x} \\ p_1 \\ \vdots \\ p_{n-d} \end{pmatrix} \end{aligned} \tag{3.2}$$

where $\mathbf{u}_i$ is an $n \times 1$ vector, $\mathbf{x}_i$ is the i^{th} element of $\mathbf{x}$, and q_F is a particular solution for $V_a \cdot q_F = -\mathbf{b}$. Since for each independent variable there is at least an element j of q such that $p_i = q_j$, we can assign $p_i \in [0,1]$ for $1 \leq i \leq n-d$. Let's denote P_X as the set $\{\mathbf{p} \mid \mathbf{x} \in X \wedge p_i \in [0,1] \ , 1 \leq i \leq n-d\}$. And let $L'_a(X) = \{U_a \mathbf{p} + q_F \mid \mathbf{p} \in P_X\}$. Since there may be bound on the possible values of the robot's configuration, $L_a(X)$ is not always the same as $L'_a(X)$. The lift mapping $L_a(X)$ of X is then $L_a(X) = L'_a(X) \cap \mathcal{C}$.

Let's call the union of P_X for all possible $X \subseteq \mathcal{W}$ as the augmented parameter space. We want to relate the volume of a subset of the augmented parameter space and the volume of its corresponding set in $\mathcal{C}$. It is obvious that $L_a(X) \subseteq L'_a(X)$ and hence $\mu(L_a(X)) \leq \mu(L'_a(X))$. Then, to relate $\mu(L'_a(X))$ with $\mu(P_X)$, we can use the change of variable formula in integration: $\int_{L'_a(X)} \partial q = \int_{P_X} |J(\mathbf{p})| \partial \mathbf{p}$, where J is the Jacobian matrix and $|\cdot|$ denotes determinant. Since U_a is a linear transformation, $\int_{L'_a(X)} \partial q = |U_a| \int_{P_X} \partial \mathbf{p}$. This means $\mu(L'_a(X)) = abs(|U_a|)\mu(P_X)$, where $abs(\cdot)$ denotes the absolute value. Hence, $\mu(L_a(X)) \leq K_a \mu(\mathrm{P}_X)$, where $K_a = \mathrm{abs}(|U_a|)$.

Now, we find the volume $\mu(\mathrm{P}_X)$ in terms of the volume of $X \subseteq \mathcal{W}$. We can com-

pute $\mu(\mathrm{P}_X)$ by discretizing P_X into infinitesimally small cubes. And then take the integration: $\mu(\mathrm{P}_X) = \int_{\mathbf{X}\in X}(\int_0^1(\ldots(\int_0^1 \partial p_{n-d})\ldots)\partial p_1)\partial\mathbf{X}$. Since for each independent parameters p_i, we integrate over all possible values, $(\int_0^1(\ldots(\int_0^1 \partial p_{n-d})\ldots)\partial p_1) = 1$ and $\mu(\mathrm{P}_X) = \int_{\mathbf{X}\in X}\partial\mathbf{X}$. Hence, $\mu(\mathrm{P}_X) = \mu(X)$. Combining this result and the result in the previous paragraph, we prove that $\mu(L_a(X)) \leq K_a\mu(X)$ where K_a is a positive constant value that depends only on a. $\square$.

One may be tempted to prove the above lemma by using the uniform continuity property of a continuous function from a compact set. However, notice that L_a is not even a function.

Using Theorem 3.1 and the above lemma, we can state the relation between the volume of the visibility sets of points in $\mathcal{F}$ and in $\mathcal{W}_\mathcal{F}$ more formally as,

Theorem 3.2 *Suppose n is the dimensionality of $\mathcal{C}$ and d is the dimensionality of $\mathcal{W}$. Let's denote A as the set of points a on the robot such that the projection P_a is defined as $P_a(q) = V_a \cdot q + \boldsymbol{b}$, where V_a is a $(d \times n)$-matrix with $rank(V_a) = d$ and $\boldsymbol{b}$ is a vector of d elements. Then, for any collision-free configuration q, $\mu(\mathcal{V}(q)) \leq \min_{a\in A} K_a\mu(\mathcal{V}(P_a(q)))$, where $K_a \geq 0$ is a constant value that depends on a, $\mathcal{V}(\cdot)$ is the visibility set, and $\mu(\cdot)$ is volume.*

Proof From Theorem 3.1, for any $a \in A$, $\mathcal{V}(q) \subseteq L_a(\mathcal{V}(P_a(q)))$. This means that $\mu(\mathcal{V}(q)) \leq \min_{a\in A}\mu(L_a(\mathcal{V}(P_a(q))))$. From Lemma 3.1, for any point $a \in A$, $\mu(L_a(\mathcal{V}(P_a(q)))) \leq K_a\mu(\mathcal{V}(P_a(q)))$. Combining the two results, we have the relation $\mu(\mathcal{V}(q)) \leq \min_{a\in A} K_a\mu(\mathcal{V}(P_a(q)))$. $\square$.

The above theorem gives us a bound on the size of the visibility sets of points in $\mathcal{F}$ without explicit construction of $\mathcal{F}$. This may be useful if one wants to a priori bound the number of milestones for solving a given problem, even though we do not compute it in this thesis. We will discuss more about this in Section 6.2.

3.3.3 When The Relations Hold

Now, we will elaborate on when the conditions in Theorem 3.1 and Theorem 3.2 hold. Suppose $\dim(\mathcal{W}) = d$ and $\dim(\mathcal{C}) = n$, V_a is a $(d \times n)$-matrix with

$rank(V_a) = d$ and $\mathbf{b}$ is a vector of d elements, the condition that the projection function P_a at point a of a given robot can be defined as $P_a(q) = V_a \cdot q + \mathbf{b}$ holds for,

- A free-flying rigid body robot where the point a is the center of rotation of the robot.
- An articulated robot where the position of point a are affected only by translational motion and its direction matrix has rank d. Below, we elaborate what a direction matrix is.

To see when the conditions hold for articulated robot, we will first elaborate the case of an articulated robot where all its joints are prismatic. From this result, the extension to other types of robots is straightforward. Let's first represent the kinematics of an articulated robot where all its joints are prismatic.

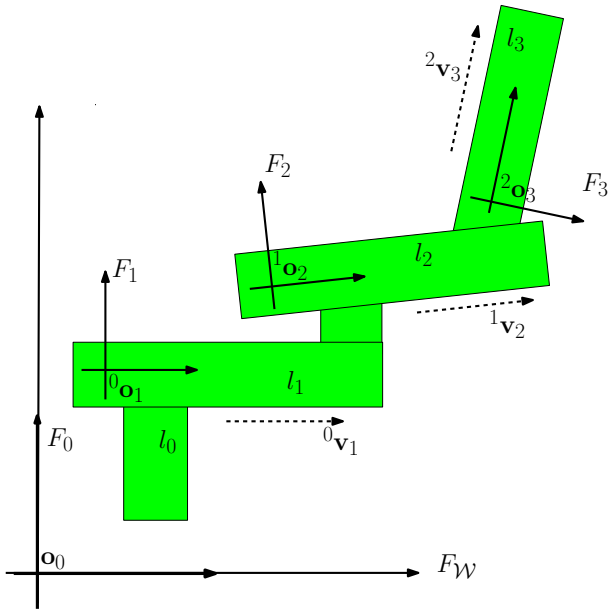


Figure 3.2: Illustration of the robot's kinematic. The robot has 3 prismatic joints and it is mounted on a static base. Each link is attached to a prismatic joint. It moves either following or opposite (depending on q_i) the direction vector ${}^{i-1}\mathbf{v}_i$, in terms of frame F_{i-1}.

Suppose robot $\mathcal{A}$ is an articulated robot with static base where all of its joints are prismatic. And suppose $\mathcal{A}$ consists of n joints and $n+1$ links, where link l_0 is the base and each link l_i ($i \in [1, n]$) is attached to a prismatic joint j_i. A (coordinate)

frame F_i is attached to each link l_i with F_0 coincides with the coordinate system of $\mathcal{W}$. Suppose a is a point in l_m and its position in F_m is denoted by $\mathbf{a}_m$. Then, the position $\mathbf{a}(q)$ of a in $\mathcal{W}$ when $\mathcal{A}$ is in configuration q, can be computed recursively as follows:

$$\mathbf{a}(q) = {}^{0}R_1\mathbf{a}_1 + {}^{0}T_1(q) \quad ; \quad \mathbf{a}_{i-1} = {}^{i-1}R_i\mathbf{a}_i + {}^{i-1}T_i(q) \ , \ i \in [2, m] \tag{3.3}$$

where $\mathbf{a}_i$ is the position of a with respect to F_i. Matrix ${}^{i-1}R_i$ is the rotation matrix that transforms the orientation of F_i to that of F_{i-1}. The rotation matrices are independent of the robot's configuration because all joints of the robot are prismatic. The vector ${}^{i-1}T_i(q)$ is the translation vector that translates the origin of F_{i-1} to coincide with that of F_i, when the robot is at configuration q. The translation vector can be computed as ${}^{i-1}T_i(q) = q_i \cdot {}^{i-1}\mathbf{v}_i + {}^{i-1}\mathbf{o}_i$, where q_i is the i^{th}-element of q, ${}^{i-1}\mathbf{v}_i$ is a normalized vector in F_{i-1} that represents the direction along which l_i will translate with respect to l_{i-1}, and ${}^{i-1}\mathbf{o}_i$ is the default position (when $q_i = 0$) of the origin of F_i in F_{i-1}. An illustration is shown in Figure 3.2.

Since in an articulated robot where all its joints are prismatic, the rotation of each coordinate frame is fixed, we want to separate the rotation from the translation part. Expanding (3.3) gives us:

$$\mathbf{a}(q) = \left(\prod_{i=1}^{m} {}^{j-1}R_j\right)\mathbf{a}_m + \sum_{i=1}^{m} q_i \left(\prod_{j=1}^{i-1} {}^{j-1}R_j\right) {}^{i-1}\mathbf{v}_i + \sum_{i=1}^{m} \left(\prod_{j=1}^{i-1} {}^{j-1}R_j\right) {}^{i-1}\mathbf{o}_i \tag{3.4}$$

and by setting:

- $\mathbf{a} = (\prod_{i=1}^{m} {}^{j-1}R_j) \cdot \mathbf{a}_m$.
- The direction vector $\mathbf{v}_i = \left(\prod_{j=1}^{i-1} {}^{j-1}R_j\right) \cdot {}^{i-1}\mathbf{v}_i$.
- $\mathbf{o}_i = \left(\prod_{j=1}^{i-1} {}^{j-1}R_j\right) \cdot {}^{i-1}\mathbf{o}_i$.

we can rewrite (3.4) as

$$\mathbf{a}(q) = \mathbf{a} + \sum_{i=1}^{m} (q_i\mathbf{v}_i + \mathbf{o}_i) \tag{3.5}$$

The direction matrix is then $V_a = [\boldsymbol{v}_1 \dots \boldsymbol{v}_m \ \mathbf{0} \dots \mathbf{0}]$, a $(d \times n)$-matrix where the first m columns correspond to $\mathbf{v}_i$ ($i \in [1, m]$) and the last $n - m$ columns are zero vectors.

Notice that the above description can be easily extended to describe the kinematics of an articulated robot with mobile base too. To represent the mobile base, we insert k (k = #dofs of the mobile base) virtual joints and links before the base. For example, suppose $\mathcal{A}$ is a planar robot with three prismatic joints and is mounted on a 2D translational mobile base. Then, we add two virtual joints and links before the base. We number the links starting from the virtual link. So, the virtual links are denoted as l_0 and l_1, the base as l_2, and the rest of the links follows the numbering. Frame F_0 is attached to l_0 and coincides with the coordinate system of $\mathcal{W}$. We assign direction vectors ${}^0\mathbf{v}_1 = [1\ 0]$ and ${}^1\mathbf{v}_2 = [0\ 1]$. And at the default position (when $q = \mathbf{0}$), we set F_0, F_1, and F_2 to coincide. Furthermore, a free-flying rigid body robot is then a special case of an articulated robot with mobile base, *i.e.*, when the robot consists of only the base.

From the above representation, it is straightforward to see that the conditions for Theorem 3.1 and Theorem 3.2, *i.e.*, a point a of the robot has linear P_a with d as the rank of the direction matrix V_a, can be satsified by any type of robot with some translational degrees of freedom, as long as the position of a is affected by translational motion where $rank(V_a) = d$.

To summarize, in this section we have shown that under certain conditions that do not depend on the number of dofs of the robot, the visibility sets of points in $\mathcal{F}$ and in $\mathcal{W}_\mathcal{F}$ are highly related. By choosing appropriate point a on the robot, the visibility sets $\mathcal{V}(q)$ of a configuration $q \in \mathcal{F}$ must lie inside the lift-mapping of the visibility set $\mathcal{V}(P_a(q))$ of its projection $P_a(q)$ in $\mathcal{W}_\mathcal{F}$. This means that to find a collision-free configuration that can be connected with a straight line segment to q, we can safely restrict our search space based on the visibility set of $P_a(q)$ in $\mathcal{W}_\mathcal{F}$. Furthermore, the volume of $\mathcal{V}(q)$ is bounded by a multiple constant of the volume of $\mathcal{V}(P_a(q))$. In Section 2.5, we have presented previous works that bound the number of milestones needed for solving a path planning problem within a certain probability of success in terms of the volume of the visibility set of $\mathcal{F}$.

Based on those results, the relation between the volume of the visibility property in $\mathcal{F}$ and $\mathcal{W}_{\mathcal{F}}$ implies that we can bound the number of milestones in terms of the volume of the visibility sets in the workspace free-space. Of course, as the number of the robot's dofs increases, the difference between the volume $\mu(\mathcal{V}(q))$ and $\mu(\mathcal{V}(P_a(q)))$ may become larger. Nevertheless, the relation that $\mathcal{V}(q)$ is a subset of $L_a(\mathcal{V}(P_a(q)))$ and the relation that the volume of $\mathcal{V}(q)$ is bounded by a multiple constant of the volume of $\mathcal{V}(P_a(q))$ holds even though the dimensionality of $\mathcal{C}$ is much higher than that of $\mathcal{W}$.

The above results and our results in Chapter 2 suggest that the size of the visibility set of the points in $\mathcal{W}_{\mathcal{F}}$ can be used to estimate a suitable sampling distribution over $\mathcal{C}$. It indicates that regardless of the dimensionality of $\mathcal{F}$, for certain types of robot, by choosing an appropriate point a on the robot, points $\mathbf{x} \in \mathcal{W}_{\mathcal{F}}$ with small visibility set corresponds to either in-collision configurations or collision-free configurations with small visibility set. It is true that we cannot guarantee that points in $\mathcal{W}_{\mathcal{F}}$ with smaller visibility sets must correspond to collision-free configurations with smaller visibility sets. But, despite the fact that $\mathcal{F}$ is unknown, the theorem implies that the size of the visibility set of a point $\mathbf{x}$ in $\mathcal{W}_{\mathcal{F}}$ gives an indication of how large the visibility sets of the configurations in $L_a(\mathbf{x})$ could be. Since a good sampling distribution samples more densely in regions of $\mathcal{F}$ with poor visibility (see Chapter 2), the results in this section strengthen our intuition that workspace information can be used to estimate a suitable sampling distribution over $\mathcal{C}$ and hence improve the performance of probabilistic path planning in solving narrow passage problem.

3.4 Workspace Importance Sampling

Now, the question is how do we use workspace information to bias sampling in the high dimensional configuration space $\mathcal{C}$. Here, we present a simple strategy for utilizing workspace information, called Workspace Importance Sampling (WIS). We present the idea and overall strategy of WIS in Section 3.4.1. We then present the details of WIS in Section 3.4.2 and Section 3.4.3.

3.4.1 Overview

WIS utilizes the relation between the visibility sets of points in $\mathcal{F}$ and in $\mathcal{W}_{\mathcal{F}}$. Intuitively, WIS uses the width of passages in $\mathcal{W}_{\mathcal{F}}$ to estimate the size of the visibility sets of points in $\mathcal{W}_{\mathcal{F}}$ and uses this estimation to generate a sampling distribution over $\mathcal{C}$.

To see why the width of passages in $\mathcal{W}_{\mathcal{F}}$ estimates the size of the visibility set, we first need to relax the requirement in Theorem 3.2, such that the relation holds for any types of robots. Since in general, rotating the robot implies that each point of the robot traverses a curve in $\mathcal{W}_{\mathcal{F}}$, we cannot guarantee that if $q' \in \mathcal{V}(q)$, then $P_a(q') \in \mathcal{V}(P_a(q))$. This means that Theorem 3.2 does not hold for robots without translational degrees of freedom, such as industrial robots where all its joints are rotational and is mounted on a static base.

Nevertheless, if we restrict the visibility set $\mathcal{V}(q)$ of a configuration q to only those configurations within a small distance from q, we can still use the geometric property of $\mathcal{W}_{\mathcal{F}}$ to roughly estimate the visibility property of $\mathcal{F}$. To estimate the visibility property of $\mathcal{F}$, we will use the distance function as defined in Section 3.2. Using this definition of distance, the set of points $\mathcal{B}(q, \Delta_{\mathcal{O}}(q))$ within $\Delta_{\mathcal{O}}(q)$ distance from q must lie in the visibility set of q. This implies that if $\Delta_{\mathcal{O}}(q)$ is large, then $\mathcal{V}(q)$ must be large as well.

Further observation on the geometry of $\mathcal{W}$ provides us with an upper bound on the distance $\Delta_{\mathcal{O}}(q)$ between a configuration and the forbidden region. This bound can be used as a rough estimation on how likely the configurations corresponding to a subset of $\mathcal{W}_{\mathcal{F}}$ have large visibility set. To identify this geometric property, we first introduce some definitions regarding the geometric property of $\mathcal{W}_{\mathcal{F}}$. We can view $\mathcal{W}_{\mathcal{F}}$ as a union of medial balls, where a medial ball is a ball that tangentially touches at least two points on the boundary of $\mathcal{W}_{\mathcal{F}}$.

Definition 3.1 *A medial region $B_r \subseteq \mathcal{W}_{\mathcal{F}}$ with radius r is a union of medial balls with radius larger than or equal to r in $\mathcal{W}_{\mathcal{F}}$. Notice that the radius of a medial region can be considered as the width of the narrowest passage covered by the medial region.*

An illustration of a medial region is shown in Figure 3.1. Now, if B_r is the smallest medial region such that all points of the robot at configuration q are enclosed in B_r, then $\Delta_{\mathcal{O}}(q) \leq 2r$. A medial region with large r indicates higher possibilities that there are configurations $q' \in L_a(B_r)$ far away from the forbidden region, and hence have large visibility set. This observation and our results in Chapter 2 suggest that we can use the radius of the medial balls of subsets of $\mathcal{W}_{\mathcal{F}}$ to roughly estimate a suitable sampling distribution over $\mathcal{C}$, regardless of the type of the robot.

Now, we describe our simple planner, workspace importance sampling (WIS). Although the radius of medial balls in $\mathcal{W}_{\mathcal{F}}$ can be used as a rough estimation of the visibility property of $\mathcal{F}$, and hence as a rough estimation of a suitable sampling distribution over $\mathcal{C}$, $\mathcal{W}_{\mathcal{F}}$ contains infinitely many medial balls that intersects one another. So, to simplify estimating the medial balls and to simplify computing the sampling distribution, WIS partitions $\mathcal{W}_{\mathcal{F}}$ into triangles. And uses the height of each triangle to estimate the radius of the smallest medial region that covers the triangle. For simplicity in writing, we use triangles in a general sense, referring to triangles in 2D workspace and tetrahedra in 3D workspace. Next, to generate a sampling distribution where smaller density are given to regions with larger visibility set, WIS assigns a sampling distribution inversely proportional to the weight which is computed based on the height of the triangle.

WIS is based on the standard multi-query PRM approach as described in Section 2.2. The overall strategy of WIS is presented in Algorithm 3.1. The details of the algorithm are presented below. WIS stops once all the given queries Q have been solved or the roadmap reaches a maximum number of milestones N. If all the given queries are solved, WIS returns a list of paths Ψ where each element is a path between a query of Q. The details of the primitive `FreeConf` for checking whether a configuration is collision free or not and the primitive `AllSolved`() for checking whether all queries have been solved or not, are presented in Appendix A.

3.4.2 Extracting Workspace Information

WIS extracts workspace information once, before roadmap construction starts. It uses this information to generate sampling distribution over $\mathcal{C}$ that will be used in

Algorithm 3.1 WIS(Q, N)

1: Let a be the robot's feature point.
2: Let $\mathcal{W}_\mathcal{F}$ be the workspace free-space, *i.e.*, subset of $\mathcal{W}$ that is not occupied by obstacles.
3: Compute a triangulation $\mathcal{T}_\mathcal{F}$ of P in $\mathcal{W}_\mathcal{F}$.
4: **for all** triangle $t \in \mathcal{T}_\mathcal{F}$ **do**
5: Calculate the weight $h(t)$ and probability $p_T(t)$ of t.
6: **repeat**
7: Sample a configuration q based on the probability distribution p_T defined over $\mathcal{T}_\mathcal{F}$.
8: **if** `FreeConf`(q) is true **then**
9: Insert q to the roadmap $\mathcal{R}$.
10: **until** `AllSolved`(Q, $\mathcal{R}$, Ψ) is true or $\mathcal{R}$ contains N milestones.
11: Return Ψ.

roadmap construction. Although this process is run only once, we would like to do it efficiently such that the total time, *i.e.*, the time for extracting workspace information and constructing the roadmap, is at least comparable to other probabilistic path planners.

Space partitioning

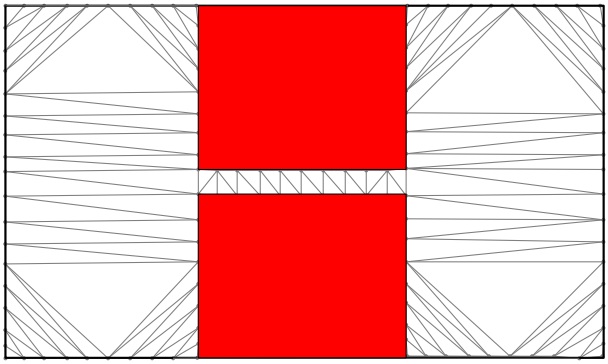


Figure 3.4: A Delaunay triangulation of $\mathcal{W}_\mathcal{F}$ in a 2-D workspace. Obstacles are colored red.

Suppose the workspace free-space $\mathcal{W}_\mathcal{F}$ is the subset of the workspace $\mathcal{W}$ that is not occupied by obstacles. The first step of WIS is to triangulate $\mathcal{W}_\mathcal{F}$ by treating $\mathcal{W}_\mathcal{F}$ as a polygon for 2D workspace and as a polyhedron for 3D workspace. However, it is known that not all polyhedra can be tetrahedralized [Toussaint et al., 1993]. To avoid this difficulty, we sample points at a fixed resolution on

the boundary of $\mathcal{W}_{\mathcal{F}}$, using an algorithm similar to scan conversion in computer graphics [Foley et al., 1995], and compute a Delaunay triangulation $\mathcal{T}$ over the set of sampled points. See Figure 3.4 for an example in a 2D workspace. If the sampling resolution is sufficiently high, then under reasonable geometric assumptions, $\mathcal{T}$ is conformal in the sense that every face on the boundary of $\mathcal{W}_{\mathcal{F}}$ is a union of faces in $\mathcal{T}$ [Amenta et al., 2001], and hence every triangle in $\mathcal{T}$ is in either $\mathcal{W}_{\mathcal{F}}$ or its complement. We will use $\mathcal{T}_{\mathcal{F}}$ to denote the subset of all triangles in $\mathcal{W}_{\mathcal{F}}$.

Weight

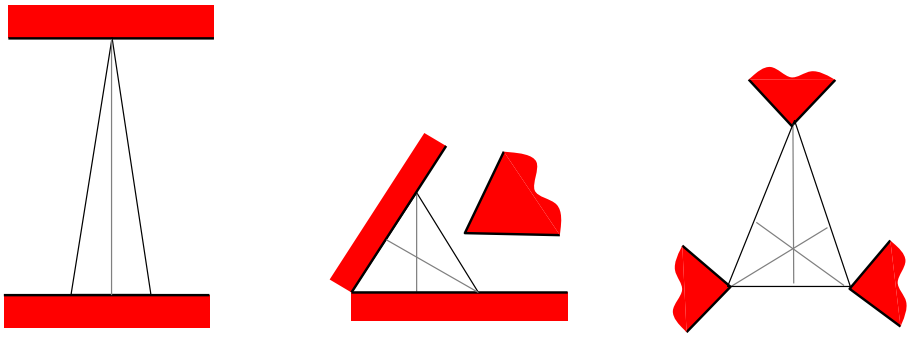


Figure 3.5: Defining the weight $h(t)$ of a triangle t in a 2-D workspace. There are three cases: (a) one edge, (b) two edges, and (c) no edge on the boundary of $\mathcal{W}_{\mathcal{F}}$.

After the triangles $\mathcal{T}_{\mathcal{F}}$ have been generated, we assign a weight $h(t)$ to every triangle t in $\mathcal{T}_{\mathcal{F}}$. Ideally, we would like to assign weight based on the relative size of the radius of the smallest medial region that encloses t. As discussed in Section 3.4.1, larger radius of medial region indicates higher possibilities that the corresponding region in $\mathcal{F}$ is wide-open and to generate a desired sampling distribution, we would like to assign lower sampling distribution to such regions. However, computing the radius of the medial region exactly, especially in 3D workspace, is complicated. On the other hand, intuitively what we want is just the width of the passage bounded by two or more "walls". Therefore, to simplify computation, we use the heights where the base coincide with the boundary of an obstacle to define the weight of the triangle. In particular, the weight $h(t)$ of a triangle t is computed as the average of the heights where the base coincide with the boundary of an obstacle.

Let's now see the computation of $h(t)$ in 3D workspace. A tetrahedron t has four heights h_i for $i = 1, 2, 3, 4$, each corresponding to a face of t. Only those heights that give an estimate of the local "width" of $\mathcal{F}$ are relevant. We thus define the weight $h(t)$ as follows:

- If t has one or more faces lying on the boundary of $\mathcal{W}_{\mathcal{F}}$, then

$$h(t) = \frac{\sum_{i=1}^{4} \beta_i h_i}{\sum_{i=1}^{4} \beta_i},$$

  where β_i is 1 if f_i lies on the boundary of $\mathcal{W}_{\mathcal{F}}$ and 0 otherwise.

- If t has none of its faces lying on the boundary of $\mathcal{W}_{\mathcal{F}}$, then $h(t) = \sum_{i=1}^{4} h_i/4$.

See Figure 3.5 for illustrations of the corresponding definition in 2D workspace. A small $h(t)$ indicates that t is likely to lie in a region of $\mathcal{W}_{\mathcal{F}}$ which is surrounded by obstacles that lie close to each other. This suggests that t corresponds to regions of $\mathcal{F}$ that are surrounded by forbidden regions that lie close to each other, and hence lie in the narrow passages of $\mathcal{F}$. So, more effort is needed to sample the region that corresponds to triangles with small weight. According to this definition of weight, a skinny triangle t, like the one shown in Figure 3.5(a), has a large value of $h(t)$. This indicates that t is not inside a narrow passage.

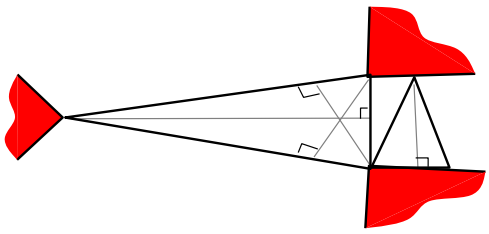


Figure 3.6: An illustration of false positive case that occurs when the minimum of the relevant height is used. When minimum height is used, the weight of the large triangle at the mouth of the narrow passage is more than the weight of the triangle inside the narrow passage. This implies that WIS assigns higher sampling distribution to region of $\mathcal{C}$ that corresponds to the large triangle, compared to region of $\mathcal{C}$ that corresponds to the triangle inside the narrow passage. However, when the average height is used, the weight of the large triangle is increased because the other relevant heights are taken into consideration. As a result, this type of false positive cases can be alleviated.

One may want to compute the weight using the minimum of the "relevant" heights instead of the average because using the minimum is less prone to false

negative cases. False negative cases occur when the weight $h(t)$ is large but t corresponds to narrow passages in $\mathcal{F}$. Since the average height is more than or equal to the minimum height, using minimum height tends to be less prone to false negative cases compared to using average height. However, this reduced false negative cases come at a cost of increased false positive cases. Figure 3.6 shows an illustration. In real world application, the case illustrated in Figure 3.6 is not rare. It occurs for instance when a robot needs to pass through a narrow door leading to a large room. False positive cases cause the planner to oversample wide-open space and hence degrade the overall performance of the planner. So, in trying to strike a balance such that the overall planner works well in many real-world problems, we prefer to use the average of all the relevant heights as the weight.

Probability

To approximate our desired sampling distribution (discussed in Section 2.5), WIS assigns probability to each triangle t in $\mathcal{T}_{\mathcal{F}}$ inversely proportional to their $h(t)$ value, as follows:

$$p_T(t) = \frac{1/h(t)}{\sum_{i=1}^{K} 1/h(t_i)} \tag{3.6}$$

where T is the random variable that represents the triangle picked from $\mathcal{T}_{\mathcal{F}}$ and K is the number of triangles in $\mathcal{T}_{\mathcal{F}}$. As discussed earlier, lower $h(t)$ is likely to indicate that t corresponds to narrow passages regions in $\mathcal{C}$. In Section 2.5 we have discussed that the desired sampling distribution assigns higher distribution to narrow passage regions and lower distribution to wide-open regions. Therefore, to approximate the desired sampling distribution, WIS assigns higher probability to triangles with lower $h(t)$ and lower probability to triangles with higher $h(t)$ and then uses this probability distribution to bias sampling in $\mathcal{C}$.

3.4.3 Sampling

Now that we have defined a probability distribution over the triangles in $\mathcal{T}_{\mathcal{F}}$, the question is how do we use this distribution to bias sampling in the high dimensional

configuration space. For this, WIS samples a configuration in two stages. First, it samples a triangle t according to probability distribution defined in (3.6).Once a triangle t is sampled, WIS samples a point $\mathbf{x}$ uniformly at random from t. Next, WIS samples a configuration uniformly at random from $L_a(\mathbf{x})$, where a is the robot's feature point. A robot's feature point is a point on the robot, chosen a priori before sampling starts. WIS represents the robot with a single feature point.

The details on how exactly WIS samples a configuration depends on the specifics of the robot's kinematics and are described below separately for rigid and articulated robots.

Rigid body robot

As a heuristic, WIS uses the centroid of the robot as the feature point for a rigid body robot, and denote this point as a.

The configuration q of a rigid body robot consists of a positional component q_τ, which specifies the position of the robot's reference point in the workspace, and an orientational component q_θ, which specifies the orientation of the robot. To sample a configuration, WIS first samples q_θ uniformly at random. In 3D workspace, q_θ is sampled uniformly at random from a unit quaternion space [Shoemake, 1985]. WIS will then sample a point $\mathbf{x} \in \mathcal{W}_\mathcal{F}$ as describe above. Finally, it computes q_τ such that at $q = (q_\tau, q_\theta)$ places the robot's feature point a at q_τ.

Articulated robot

The configuration $q = (q_1, \ldots, q_n)$ of an articulated robot specifies its joints parameters. For an articulated robot with static base, WIS uses the wrist point, *i.e.*, the centroid of the wrist, as the robot's feature point. Suppose $q_1, \ldots, q_m$ ($m \leq n$) determines the position of the wrist point. WIS starts sampling $\mathcal{F}$ by sampling a point $\mathbf{x}$ from $\mathcal{W}_\mathcal{F}$ according to the density function in (3.7). It will then find $q_1, \ldots, q_m$ by solving the robot's inverse kinematics (IK) equations. If IK has no solution, WIS samples another point until it finds a point where IK returns one or more solutions. If IK returns more than one solution, WIS selects one of them at random. Various improvements can be made to speed-up this process.

For instance, we may restrict the sampling domain according to the reachability of the wrist point. WIS will then sample the other joint parameters $q_{m+1}, \ldots, q_n$ uniformly at random.

When the articulated robot is mounted on a mobile base, WIS uses the origin of the base frame as the feature point. WIS samples the position and orientation of the base exactly as it samples the position and orientation of the rigid body robot. Then, for the rest of the degrees of freedom, WIS samples each degree of freedom uniformly at random.

3.5 Implementation and Experiments of WIS

We implemented WIS using C++ and using the Qhull library [Barber et al., 1996] for triangulation. To triangulate $\mathcal{W}_{\mathcal{F}}$ of a workspace, we a priori determine the resolution for sampling the boundary of $\mathcal{W}_{\mathcal{F}}$. We set the resolution to be high enough so that under reasonable geometric assumptions, the resulting triangulation is conformal. Implementation details of the primitives, *i.e.*, `FreeConf`, `FreePath`, and `AllSolved` are presented in Appendix A.

For comparison, we implemented the basic-PRM [Kavraki et al., 1996b], Generalized Voronoi Graph (GVG) [Foskey et al., 2001], and the original Adaptive Hybrid Sampling (AHS) [Hsu et al., 2005] which combines uniform distribution, several Gaussian strategy [Boor et al., 1999] with different parameters, and several randomized bridge builder (RBB) [Sun et al., 2005] with different parameters. The basic-PRM is used as a benchmark of the difficulty of the problem. GVG is used because it is one of the workspace-based probabilistic path planner that has been proposed. GVG uses the Generalized Voronoi Graph of the workspace free-space to find a path in the workspace and then modify this path to find a valid path in $\mathcal{F}$. A more elaborate explanation of GVG can be seen in Section 2.6.1. We only implemented GVG for 2D workspace because constructing the Generalized Voronoid Graph for 3D workspace is significantly more complicated than constructing it for 2D workspace and the results of GVG in 2D workspace is already discouraging. The original-AHS is used because it is one of the recent probabilistic path planners

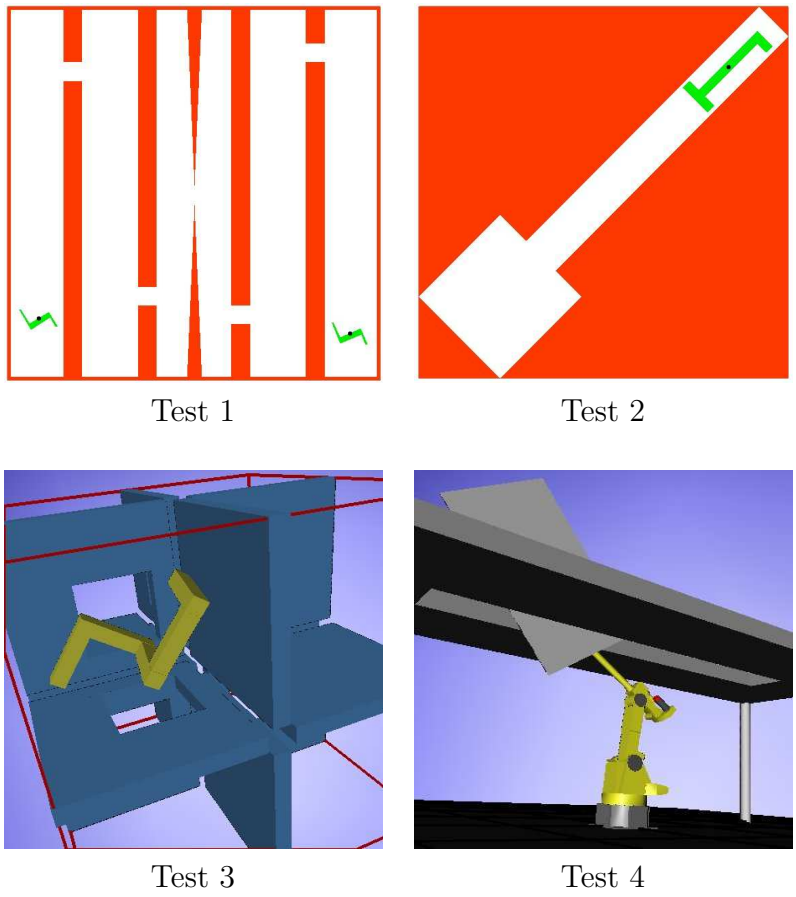


Figure 3.7: Test scenarios. **Test 1**: A 3-dofs rigid-body robot moves from the lower left corner to the lower right corner by passing through five narrow openings. **Test 2**: A 3-dofs rigid-body robot turns 180 degrees in a narrow dead-end. **Test 3**: A 6-dofs rigid-body robot must pass through 6 out of 7 narrow openings in order to answer the given query. **Test 4**: A 6-dofs robot manipulator with its end-effector holding a large plate maneuvers through a narrow slot.

that has been shown to perform well. Furthermore, its component samplers, *i.e.*, Gaussian and RBB, have been shown to perform well and its combination have been shown to perform even better than each of its component sampler alone.

We tested WIS on several scenarios involving 2D and 3D workspace as well as rigid and articulated robot. The scenarios are shown in Figure 3.7. To set the parameters of each planner, we select several representative values for the parameter(s). Each planner with different parameter values were run 10 times independently on each test scenario. The parameter values that generate the best performance are then used as the parameter for testing the planner on the particular scenario. For testing, each planner were run 30 times independently on each

Table 3.1: Performance comparison of several probabilistic path planners. All times are measured in seconds.

Planner	**Test 1**			
	T_{pre}	$T_{\text{tot}} \pm$ std	N_{mil}	N_{sam}
Basic-PRM		75.9 ± 45.1	13,540	52,687
Original AHS		23.0 ± 8.5	3,477	164,776
GVG	0.027	49.5 ± 22.6	11,548	61,653
WIS	0.034	6.7 ± 2.0	1,660	7,024
Planner	**Test 2**			
	T_{pre}	$T_{\text{tot}} \pm$ std	N_{mil}	N_{sam}
Basic-PRM		4.1 ± 1.3	601	53,616
Original AHS		3.3 ± 1.3	163	76,742
GVG	0.007	6.2 ± 1.2	693	74,289
WIS	0.007	0.7 ± 0.3	154	11,521
Planner	**Test 3**			
	T_{pre}	$T_{\text{tot}} \pm$ std	N_{mil}	N_{sam}
Basic-PRM		94.6 ± 48.6	9,011	36,594
Original AHS		56.7 ± 20.2	1,669	198,313
WIS	0.607	80.3 ± 30.0	5,723	160,686
Planner	**Test 4**			
	T_{pre}	$T_{\text{tot}} \pm$ std	N_{mil}	N_{sam}
Basic-PRM		69.8 ± 26.0	9,246	35,878
Original AHS		56.0 ± 27.9	2,672	168,013
WIS	0.071	200.7 ± 90.4	14,423	961,613

T_{pre}: time for triangulating$\mathcal{W}_{\mathcal{F}}$.
$T_{\text{tot}} \pm$ std: total running time ± standard deviation.
N_{mil}: number of milestones required for answering the query.
N_{sam}: number of configurations sampled.

testing scenario. All experiments were conducted on a PC with a 3 GHz Intel Pentium 4 and 1 GB memory.

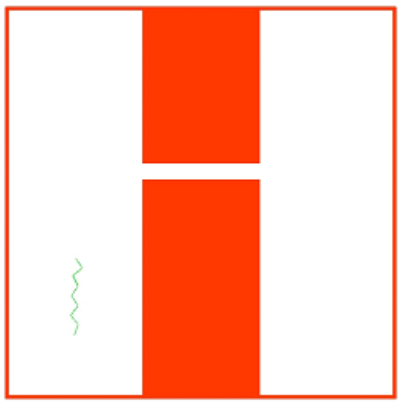
The average results of the 30 runs are shown in Table 3.1. The results show that in the first two scenarios, WIS performs 3 to 5 times faster than the original-AHS and 6 to 11 times faster than the basic-PRM. WIS uses fewer milestones compared to the basic-PRM and the original-AHS. Which indicate that WIS is able to place milestones at more important regions of $\mathcal{F}$.

It is interesting to note that the previous workspace-based probabilistic path planner GVG does not perform well in both **Test 1** and **Test 2**. In **Test 1**, GVG

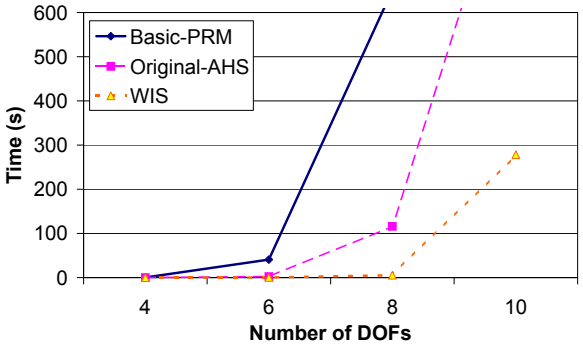
performs poorly, even though the workspace path it predicts is quite close to the path traversed by the robot when moving from the initial to the goal configurations. The reason is that at the narrow openings, the robot needs to wiggle a lot and needs to move a little outside the narrow openings and their "mouths". So, the path generated in $\mathcal{F}$, by GVG's local sampling, is disconnected at the narrow openings. Therefore, to connect the disconnected path, GVG needs to run EST [Hsu et al., 1999] over the whole configuration space. However, since the main objective of EST is to explore the sampling domain as fast as possible, instead of sampling densely near and inside the narrow openings, EST spreads its samples throughout the whole configuration space. This causes GVG to over-sample wide-open regions of $\mathcal{F}$, and hence slows down the overall performance of the planner significantly.

In **Test 2**, the initial and goal configuration q_i and q_g, when projected to $\mathcal{W}$, are very close. However, to go from q_i to q_g, the robot must go out of the narrow tunnel, reorient, and then go back to the tunnel again. This scenario may potentially mislead planners that use workspace path to find a valid path in $\mathcal{F}$. And as we can see in Table 3.1, GVG is mislead by the workspace information. The reason is that by first predicting a path between the initial and goal position of the robot using the Generalized Voronoi Graph of the workspace free-space, GVG predicts that the path is the short path inside the narrow tunnel. As a result, GVG spends a significant amount of time trying to construct a valid path in $\mathcal{F}$ based on the predicted workspace path without success. And in the end, GVG has to revert back to EST with the whole $\mathcal{C}$ as the sampling domain in order to find a valid path in $\mathcal{F}$. Notice however that this scenario does not mislead all workspace-based sampling strategies. As we can see in Table 3.1, WIS is not fooled by this scenario even though WIS uses workspace information because WIS does not use workspace path to predict a path between the initial and goal configurations.

However, in **Test 3**, WIS performs 1.5 times slower than the original-AHS and only slightly faster than the basic-PRM. In this environment, the triangulation generates many "short" triangles where its lift mapping lies in the forbidden region of $\mathcal{C}$. Hence, WIS wastes a lot of time for oversampling some parts of the forbidden region of $\mathcal{C}$ without gaining much. As a result, although the number



Test 5



Results

Figure 3.8: Additional test for testing the performance of the planners as dim($\mathcal{C}$) increases.

of milestones needed by WIS is only half of that needed by the basic-PRM, the overall performance of WIS is only slightly better than that of the basic-PRM. Nevertheless, compared to the original-AHS, WIS needs more than 3 times more milestones in order to solve the problem. This is not surprising, as the original-AHS uses machine learning technique to generate adaptive sampling distribution. It is able to identify regions of $\mathcal{F}$ that have been oversampled, and hence reduce the sampling density on these regions. Therefore, the original-AHS does not waste too much time for oversampling a well represented region.

In **Test 4**, WIS performs around 3 times slower than both the basic-PRM and the original-AHS. With only the robot's wrist point representing the whole robot, there are only very few triangles where its lift mapping intersects the narrow passage region. Furthermore, in this case, the boundaries of the workspace are near to the bars. So, there are many short triangles that do not lie in the narrow slot. Since most of these short triangles correspond to wide-open space in $\mathcal{F}$ and since WIS favors sampling from these triangles as much as sampling from the short triangles in the narrow slot, WIS oversamples wide-open space of $\mathcal{F}$ and generates many unnecessary milestones there.

One may suspect that WIS performs poorly in **Test 3** and **Test 4** because the usefulness of workspace information diminishes as dim($\mathcal{C}$) becomes much more

than $\dim(\mathcal{W})$. To test this, we ran additional tests, *i.e.*, **Test 5** (Figure 3.8). In this test, the robot is a planar articulated arm with a free-flying base. The dimensionality of $\mathcal{C}$ is increased by adding up to 8 links to the robot, resulting in a maximum of 10 dofs. The robot must move through the narrow passage in the middle. The results are shown in Figure 3.8. And it clearly indicates that workspace information still has its merit. As $\dim(\mathcal{C})$ increases, the decrement of the performance of WIS is slower than that of the basic-PRM or the original-AHS. This result strengthens our argument in Section 3.3 that workspace information provides useful information for generating suitable sampling distributions for probabilistic path planning, regardless of the difference between the dimensionality of $\mathcal{C}$ and the dimensionality of $\mathcal{W}$.

3.6 Analysis of WIS

In this section, we will analyze the probabilistic completeness and the running time of WIS. We start by deriving the sampling density of WIS in Section 3.6.1. We then present the probabilistic completeness and running time of WIS in Section 3.6.2 and Section 3.6.3, respectively.

3.6.1 Sampling Density

To derive the sampling density of WIS, let's look at the two-stage sampling procedure of WIS along with the sampling density in each stage. In the first stage, WIS samples a point $\mathbf{x}$ in $\mathcal{W}_{\mathcal{F}}$ by first sampling a triangle t from $\mathcal{T}_{\mathcal{F}}$ based on the distribution $p_T(t)$ as defined in (3.6). It will then sample a point $\mathbf{x}$ uniformly at random from t. So, the density function $f_X(\mathbf{x})$ used by WIS to sample $\mathbf{x} \in \mathcal{W}_{\mathcal{F}}$ can be computed as

$$\begin{aligned} f_X(\mathbf{x}) &= \sum_{t \in \mathcal{T}_{\mathcal{F}}} f_{X|T}(\mathbf{x} \mid t) p_T(t) \quad ; \quad f_{X|T}(\mathbf{x} \mid t) = \begin{cases} \frac{1}{\mu(t)} & \mathbf{x} \in t \ , \ t \in \mathcal{T}_{\mathcal{F}} \\ 0 & \text{otherwise} \end{cases} \\ &= \frac{p_T(\tau(\mathbf{x}))}{\mu(\tau(\mathbf{x}))} \end{aligned} \tag{3.7}$$

where X and T are the random variables that represent a point sampled from $\mathcal{W}_\mathcal{F}$ and a triangle sampled from $\mathcal{T}_\mathcal{F}$, $\mu(\cdot)$ denotes volume, and $\tau(\mathbf{x})$ is the triangle that contains $\mathbf{x}$.

In the second stage, after a point $\mathbf{x} \in \mathcal{W}_\mathcal{F}$ is sampled, a configuration is sampled uniformly at random from $L_a(\mathbf{x})$. So, if X and Q are the random variables that represent the point in $\mathcal{W}_\mathcal{F}$ and the configuration sampled by WIS, then the probability density function $f_{Q|X}(q \mid \mathbf{x})$ of sampling q given $\mathbf{x}$ is defined as

$$f_{Q|X}(q \mid \mathbf{x}) = \begin{cases} \frac{1}{\lambda(L_a(\mathbf{x}))} & q \in L_a(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

where $\lambda(\cdot)$ denotes the the volume in the $(n - \dim(\mathcal{W}))$-dimensional subspace of $\mathcal{C}$. Assuming that $\mathcal{C}$ is a normalized Euclidean space $[0,1]^{\dim(\mathcal{C})}$, the largest possible $\lambda(L_a(\mathbf{x}))$ is one.

The probability density function $f_Q(q)$ generated by WIS can then be computed as

$$\begin{aligned} f_Q(q) &= \int_{\mathbf{x} \in \mathcal{W}_\mathcal{F}} f_{Q|X}(q \mid \mathbf{x}) f_X(\mathbf{x}) \partial \mathbf{x} \\ &= \frac{1}{\lambda(L_a(P_a(q)))} \frac{p_T(\tau(P_a(q)))}{\mu(\tau(P_a(q)))} \end{aligned} \tag{3.9}$$

3.6.2 Probabilistic Completeness

Once we have the sampling density, it is straightforward to analyze WIS using any analysis of the basic-PRM, *e.g.*, the analysis in [Kavraki et al., 1995, Kavraki et al., 1996a, Hsu et al., 1997], to analyze WIS. The only modification required is to replace the uniform distribution with the distribution generated by WIS. Below, we show an example of adapting one of the analysis of the basic-PRM, *i.e.*, the one in [Kavraki et al., 1996a], to analyze WIS. The result shows that the probability that WIS solves any given queries converges to one, exponentially in terms of the number of milestones, provided such a solution exists. More formally,

Theorem 3.3 *Let ψ be a collision-free path that solves the given query. Suppose l is the length of ψ, d is the shortest distance between a configuration in ψ to*

the forbidden region, and $\mathcal{B}_{\frac{d}{2}}$ *is any ball of radius* $\frac{d}{2}$ *in* $\mathcal{F}$*. Let's denote* Q_ψ *as the union of all balls of radius* $\frac{d}{2}$ *centered at each configuration in* ψ *and* $\tau'(\psi) = \{t \mid t \in \mathcal{T}_\mathcal{F}, t \bigcap P_a(Q_\psi) \neq \emptyset\}$*. Then, the probability that* ψ *cannot be found using a roadmap of* N *milestones that has been generated by WIS is* $Pr(failure) \leq \left(\lceil \frac{2l}{d} \rceil - 1\right)(1 - (K_{\mathcal{T}'(\psi)} \cdot \mu(\mathcal{B}_{\frac{d}{2}})))^N$*, where* $K_{\mathcal{T}'(\psi)} = \min_{t \in \mathcal{T}'(\psi)} \frac{p_\mathcal{T}(t)}{\mu(t)}$ *and* $K_{\mathcal{T}'(\psi)} \geq \epsilon$ *for* $\epsilon > 0$*.*

Proof Suppose $B = \lceil \frac{2l}{d} \rceil$. We can discretize ψ into a sequence of configurations $q^0, \ldots, q^B$ such that for $k \in [0, B]$, the length of ψ between q^k and q^{k+1} is at most $\frac{d}{2}$, $q^0 = q_i$, and $q^B = q_g$. Then, ψ can be covered with balls $\mathcal{B}(q^k, \frac{d}{2})$ of radius $\frac{d}{2}$ and center at q^k. An illustration is shown in Figure 3.9. Notice that $\mathcal{B}(q^{k+1}, \frac{d}{2}) \subseteq \mathcal{B}(q^k, d)$. Hence, if the roadmap constructed by WIS places at least

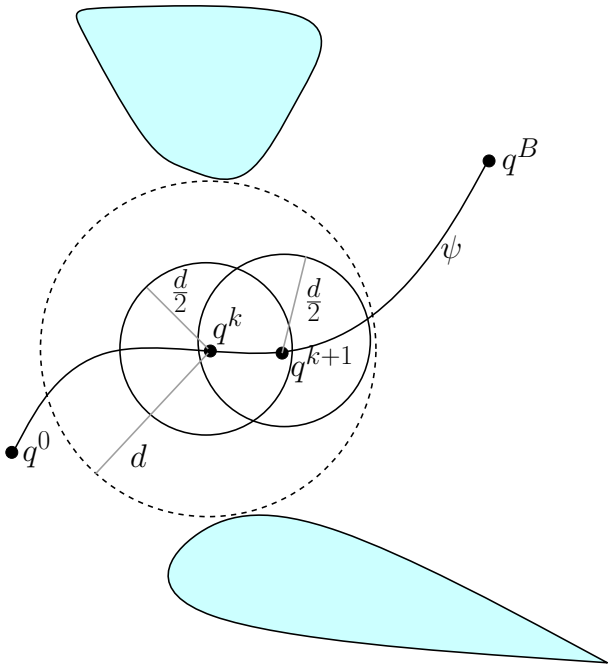


Figure 3.9: Illustration of the path ψ and the discretization of the path. The forbidden region is colored light green.

one milestone in each ball, then the generated roadmap finds the path. So, the probability that WIS fails to find a path after the generated roadmap contains N milestones can be written as,

$$Pr(failure) \leq Pr[\exists k \; \mathcal{B}(q^k, \tfrac{d}{2}) \text{ is empty}]$$

$$\begin{aligned} &\leq \sum_{k=1}^{B-1} Pr[\mathcal{B}(q^k, \tfrac{d}{2}) \text{ is empty}] \\ &= \sum_{k=1}^{B-1} (1 - Pr[q \in \mathcal{B}(q^k, \tfrac{d}{2})])^N \\ &= \sum_{k=1}^{B-1} \left(1 - \int_{q \in \mathcal{B}(q^k, \frac{d}{2})} f_Q(q) \partial q\right)^N \end{aligned}$$

Substituting (3.9) to the equation, the right hand side of the equation becomes:

$$= \sum_{k=1}^{B-1} \left(1 - \int_{q \in \mathcal{B}(q^k, \frac{d}{2})} \frac{1}{\lambda(L_a(P_a(q)))} \frac{p_T(\tau(P_a(q)))}{\mu(\tau(P_a(q)))}\right)^N$$

Since q must lie in $L_a(\mathbf{x})$ and $\mathbf{x}$ in t, we can rewrite the above equation as

$$\begin{aligned} &= \sum_{k=1}^{B-1} \left(1 - \sum_{t \in \mathcal{T}_\mathcal{F}} \int_{\mathbf{x} \in t} \int_{q \in \left(\mathcal{B}(q^k, \frac{d}{2}) \cap L_a(\mathbf{x})\right)} \frac{1}{\lambda(L_a(\mathbf{x}))} \frac{p_T(t)}{\mu(t)} \partial q \partial \mathbf{x}\right)^N \\ &= \sum_{k=1}^{B-1} \left(1 - \sum_{t \in \mathcal{T}_\mathcal{F}} \int_{\mathbf{x} \in t} \frac{\lambda(\mathcal{B}(q^k, \frac{d}{2}) \bigcap L_a(\mathbf{x}))}{\lambda(L_a(\mathbf{x}))} \frac{p_T(t)}{\mu(t)} \partial \mathbf{x}\right)^N \end{aligned}$$

By taking the largest possible value for $\lambda(L_a(\mathbf{x}))$, we have:

$$\begin{aligned} Pr(failure) &\leq \sum_{k=1}^{B-1} \left(1 - \sum_{t \in \mathcal{T}_\mathcal{F}} \int_{\mathbf{x} \in t} \lambda(\mathcal{B}(q^k, \tfrac{d}{2}) \bigcap L_a(\mathbf{x})) \frac{p_T(t)}{\mu(t)} \partial \mathbf{x}\right)^N \\ &= \sum_{i=k}^{B-1} \left(1 - \sum_{t \in \mathcal{T}_\mathcal{F}} \mu(\mathcal{B}(q^k, \tfrac{d}{2}) \bigcap L_a(t)) \frac{p_T(t)}{\mu(t)}\right)^N \\ &\leq \sum_{i=k}^{B-1} \left(1 - \min_{t \in \mathcal{T}'(\psi)} \frac{p_T(t)}{\mu(t)} \mu(\mathcal{B}(q^k, \tfrac{d}{2}))\right)^N \end{aligned}$$

Since the terms within the summation are the same for all balls $\mathcal{B}(q^k, \frac{d}{2}), k \in [1, B-1]$, we have:

$$Pr(failure) \leq \left(\left\lceil \frac{2l}{d} \right\rceil - 1\right) \left(1 - K_{\mathcal{T}'(\psi)} \cdot \mu(\mathcal{B}_{\frac{d}{2}})\right)^N$$

Which is the upper bound of the failure probability we want.

Now, we will prove that $K_{\mathcal{T}'(\psi)} \geq \epsilon$ for $\epsilon > 0$ and for any path ψ. Suppose the smallest possible height, among all possible triangles in $\mathcal{T}_\mathcal{F}$ and among all possible heights of a triangle in $\mathcal{T}_\mathcal{F}$, is ϵ_1. Since the volume of triangles in $\mathcal{T}_\mathcal{F}$ are all non-zero, $\epsilon_1 > 0$. Expanding $p_T(t)$ using (3.6), we have:

$$\begin{aligned} K_{\mathcal{T}'(\psi)} &\geq \min_{t \in \mathcal{T}_\mathcal{F}} \frac{p_T(t)}{\mu(t)} \\ &= \min_{t \in \mathcal{T}_\mathcal{F}} \frac{\frac{1}{h(t)}}{\mu(t) \sum_{i=1}^{K} \frac{1}{h(t_i)}} \end{aligned}$$

where K is the number of triangles in $\mathcal{T}_\mathcal{F}$. Since the smallest height of a triangle in $\mathcal{T}_\mathcal{F}$ is ϵ_1, $\min_{i=1}^{K} h(t_i) \geq \epsilon_1$ and we can bound the above equation as follows:

$$\min_{t \in \mathcal{T}_\mathcal{F}} \frac{p_T(t)}{\mu(t)} \geq \frac{\frac{1}{h(t)}}{\mu(t) K \frac{1}{\epsilon_1}} \tag{3.10}$$

Now, we will separate the 2D workspace case and the 3D workspace case. First, we will find a bound for the 2D workspace case. Assuming that the workspace is a Euclidean space $[0,1]^2$, $h(t) \leq \sqrt{2}$ and the largest area possible for a triangle inside the workspace is $\frac{1}{2}$. So, (3.10) becomes:

$$\min_{t \in \mathcal{T}_\mathcal{F}} \frac{p_T(t)}{\mu(t)} \geq \frac{\sqrt{2}\epsilon_1}{K}$$

Since for any edge e of a triangle t, we can construct a right triangle where e is the hypotenuse and one of t's height is a side of the right triangle, the length of any edge e of t must be more than ϵ_1. Therefore, the lower bound on the volume of t is $\mu(t) \geq \frac{\epsilon_1^2}{2}$. Assuming that the workspace is a Euclidean space $[0,1]^2$, $K \leq \frac{2}{\epsilon_1^2}$. And hence for 2D workspace, we have the following bound:

$$K_{\mathcal{T}'(\psi)} \geq \min_{t \in \mathcal{T}_\mathcal{F}} \frac{p_T(t)}{\mu(t)} \geq \frac{\epsilon_1^3}{\sqrt{2}}$$

So, by assigning $\epsilon = \frac{\epsilon_1^3}{\sqrt{2}}$, we have shown that in 2D workspace, $K_{\mathcal{T}'(\psi)} \geq \epsilon$ for $\epsilon > 0$.

The same strategy can also be used for 3D workspace. Assuming that the

workspace is a Euclidean space $[0,1]^3$, $h(t) \leq \sqrt{3}$ and the largest area possible for a triangle inside the workspace is $\frac{1}{3}$. Furthermore, since for any edge e of a tetrahedron t, we can construct a right triangle where e is the hypotenuse and one of t's height is a side of the right triangle, the length of any edge e of t must be more than ϵ_1. With similar reasoning, any height of the triangles that compose t must be more than ϵ_1 too. Therefore, the lower bound on the volume of t is $\mu(t) \geq \frac{\epsilon_1^3}{6}$. Substituting these values to (3.10) will then give us:

$$K_{\mathcal{T}'(\psi)} \geq \min_{t \in \mathcal{T}_\mathcal{F}} \frac{p_T(t)}{\mu(t)} \geq \frac{\epsilon_1^4}{2\sqrt{3}}$$

By assigning $\epsilon = \frac{\epsilon_1^4}{2\sqrt{3}}$, we have shown that in 3D workspace, $K_{\mathcal{T}'(\psi)} \geq \epsilon$ for $\epsilon > 0$.

So, we have proved that WIS is probabilistically complete. $\square$.

Compared to the basic-PRM where the upper bound of its failure probability is $\lceil \frac{2l}{d} \rceil (1 - \mu(\mathcal{B}_{\frac{d}{2}}))^N$, WIS has a lower upper bound of failure probability when $K_{\mathcal{T}'(\psi)} > 1$. When this happens, in general, the triangles in $\tau'(\psi)$ have small weights. Assuming the weight approximates the width of the passages in the workspace well, $\tau'(\psi)$ often happens when following the path ψ means that the robot moves inside narrow passages in the workspace. Hence, WIS tends to have lower failure probability than the basic-PRM when the robot moves inside narrow passages of the workspace.

3.6.3 Running Time

The total running time of WIS consists of two main parts. First, is the time for generating triangles $\mathcal{T}_\mathcal{F}$ along with its weight and probability values. This computation is performed only once before WIS starts building the roadmap. It consists of four steps. First is sampling on the boundary of $\mathcal{W}_\mathcal{F}$ using the scan-conversion like algorithm [Foley et al., 1995]. This step takes $O(|P|)$ time where $|P|$ is the number of sampled points. Second is triangulating the sampled points to generate $\mathcal{T}$. This step takes $O(|P|^2)$ in the worst case, but in practice, we can expect $O(|P| \lg |P|)$ [Amenta et al., 2001]. The third step is separating the triangles inside $\mathcal{W}_\mathcal{F}$ from those outside to generate $\mathcal{T}_\mathcal{F}$. This computation takes

$O(|\mathcal{T}|)$ where $|\mathcal{T}|$ is the number of triangles in $\mathcal{T}$. Finally, WIS assigns weight and probability values to each triangle in $\mathcal{T}_{\mathcal{F}}$. This computation takes linear time in the number of triangles in $\mathcal{T}_{\mathcal{F}}$. The number of triangles is $O(|P|)$ for 2D workspace and $O(|P|^2)$ for 3D workspace. So in the worst case, the total time T_{pre} for preprocessing the workspace information is $O(|P|^2)$.

The second part of WIS running time is for building the roadmap. This part iteratively performs two main steps. First is sampling a collision-free configuration. Let's denote the time used by WIS to generate a milestone as T_m. The second step is adding and connecting the new milestone to the milestones in the current roadmap. We denote this cost as T_l. The total time for WIS to build a roadmap of N_{mil} milestones is then $O(N_{\text{mil}} \cdot (T_m + T_l))$. So, the total running time complexity of WIS is $O(T_{\text{pre}} + N_{\text{mil}}(T_m + T_l))$.

Let's now compare the running time of WIS with that of the basic-PRM. The following method of comparing the running time of different sampling strategies have been used in [Sun et al., 2005]. Compared to the basic-PRM, WIS performs additional computation to extract workspace information. As we have seen in Section 3.5, in general this cost is very small compared to the total cost. Furthermore, to sample a configuration using information from $\mathcal{W}_{\mathcal{F}}$, WIS requires an additional constant time c. For rigid body robot, the additional time c is due to the additional computation for computing a configuration given a sampled workspace point (see Section 3.4.3). While for articulated robot, the additional time c is dominated by the inverse kinematics (IK) computation. Suppose to generate a milestone, we need N_s samples. To simplify the comparison, let's assume that the number of samples needed to generate a milestone is the same everywhere. Then, the total time T_{wis} used by WIS to generate a roadmap of N_{mil} milestones is $T_{wis} = T_{\text{pre}} + N_{\text{mil}}(N_s(T_s + c) + T_l)$, where T_s is the time used by the basic-PRM to generate a sample. While the total time T_{uni} used by the basic-PRM is $T_{uni} = N_{\text{mil}}(N_s \cdot T_s + T_l)$. However, WIS places its milestones at more important regions of $\mathcal{F}$ and hence uses less number of milestones to solve the given queries. Since in general T_l is much larger than T_s, the total time for WIS to generate a roadmap that can solve the given queries is faster

than the total time used by the basic-PRM. As an example, suppose to solve the given queries, the basic-PRM uses 3000 milestones while WIS uses 1000 milestones. And suppose $T_{\text{pre}} = 300T_s$, $N_s = 2$, $c = T_s$, and $T_l = 10T_s$. Then $T_{wis}/T_{uni} = (300 \cdot T_s + 1000 \cdot (2 \cdot 2 \cdot T_s + 10 \cdot T_s))/(3000 \cdot (2 \cdot T_s + 10 \cdot T_s))$. Which results in $T_{wis} \approx 0.4T_{uni}$. So, compared to the basic-PRM, WIS pays a slightly higher cost to place milestones at more useful regions of $\mathcal{F}$, such that the total milestones needed to solve the given queries are less and hence the total time for solving the queries are less, too.

3.7 Discussion

In this chapter, we have explored the use of workspace information to generate a suitable sampling distribution for probabilistic path planning. We have shown that under certain conditions that do not depend on the dimensionality of $\mathcal{F}$, the visibility sets of configurations in $\mathcal{F}$ can be bounded by the lift mapping of the visibility sets of the corresponding points in $\mathcal{W}_{\mathcal{F}}$. Furthermore, we have presented a simple workspace-based probabilistic path planner called workspace importance sampling (WIS). WIS efficiently extracts local geometric property of $\mathcal{W}_{\mathcal{F}}$ to estimate the size of the visibility sets of configurations in $\mathcal{F}$. The estimation is then used to assign sampling distribution over $\mathcal{C}$. In several experiments, WIS outperforms recent probabilistic path planner that has been shown to perform well. Furthermore, as the dimensionality of $\mathcal{C}$ increases, WIS's performance decreases slower than the recent probabilistic planner. Our analysis shows that the failure probability of WIS converges to zero exponentially in the number of milestones, whenever a solution exists. Furthermore, when the robot moves in a narrow region of $\mathcal{W}_{\mathcal{F}}$, WIS has a lower upper bound on the failure probability compared to the basic-PRM. These results suggest that workspace information is potentially useful for constructing a suitable sampling distribution for probabilistic path planning.

One may want to improve WIS by finding a more suitable triangulation method, as different triangulation affects the quality of the estimated width of a workspace passages, which in turn affects the constructed sampling distribution and the

overall performance of the planner. Nevertheless, regardless of the triangulation method, utilizing geometric property of the workspace alone to guide sampling in the configuration space is still prone to misleading workspace information. Therefore, instead of trying to find a more suitable triangulation, we prefer to find a better strategy to alleviate the problem of misleading workspace information.

The use of workspace information for probabilistic path planning may be misleading such that the sampling distribution constructed is unsuitable. This is expected as $\mathcal{F}$ is a convolution of both the workspace and the robot. By using only workspace information and representing the robot as a single feature point, WIS has entirely ignored the robot's geometry.

One obvious remedy is to incorporate more robot information to our sampling strategy. The main difficulty is how to incorporate more robot information and then combine it with the workspace information efficiently. We need to be careful in trying to incorporate more robot information because otherwise, we may be back trying to generate an exact representation of $\mathcal{F}$, which requires infeasible cost. A possible strategy is to represent the robot as a set of feature points instead of just a single feature point. For this, several issues need to be addressed. For instance, how many and which points of the robot should one use. And more importantly, how to efficiently combine information from multiple points of the robot with information from the workspace.

As incorporating more complicated information from the robot may be inefficient, another strategy is to directly combine workspace information with partial information about $\mathcal{F}$. Although we do not have an explicit representation of $\mathcal{F}$ and constructing it is infeasible, sampling history provides partial information about $\mathcal{F}$. We can infer local geometric property of $\mathcal{F}$ from sampling history. And workspace information will then provide a rough global information about the connectivity of $\mathcal{F}$. By combining a rough global connectivity information with a more refined local geometric information, we can infer a more detailed information about $\mathcal{F}$ better. The issue is of course how to infer information about $\mathcal{F}$ from the sampling history and how to efficiently combine this information with the workspace information.

In Chapter 2, we have argued that a sampling strategy that dynamically adapt

its sampling distribution is more suitable for probabilistic path planning. In probabilistic path planning, sampling distribution represents the planner's belief on which part of $\mathcal{C}$ is more useful to improve its current understanding about the shape of $\mathcal{C}$. And of course the use of sampling different parts of $\mathcal{C}$ changes as the roadmap construction progresses. So, it seems more suitable if the sampling distribution is dynamic. The main difficulty in generating dynamic sampling distribution is of course how should one adapt the sampling distribution. Previous probabilistic path planners that generate dynamic sampling distribution use sampling history to dynamically adapt its sampling distribution. However, to identify narrow passages and sample more densely there, the planner needs to first sample configurations from inside or around the narrow passages. This is difficult to do and may require significant amount of time before the planner can bias sampling toward narrow passages. Since workspace information provides a rough global information about $\mathcal{F}$, it may be useful to use workspace information to adapt sampling distribution. However, workspace information has only been used for non-adaptive sampling. The main reason is that to use workspace information for adaptive sampling, the strategy needs to repeatedly map information from and to the robot's free-space $\mathcal{F}$ and the workspace free-space $\mathcal{W}_{\mathcal{F}}$. Since $\mathcal{F}$ and $\mathcal{W}_{\mathcal{F}}$ are two distinct spaces, this mapping is often considered as expensive. However, the strong relation between $\mathcal{F}$ and $\mathcal{W}_{\mathcal{F}}$ and the small workspace extraction time shown in this chapter indicate that mapping information from and to $\mathcal{F}$ and $\mathcal{W}_{\mathcal{F}}$ can be done efficiently. And therefore it would be possible to use workspace information for adaptive sampling.

In the next chapter, we present a workspace-based probabilistic path planner that addresses the above issues.

Chapter 4

Workspace Information for Adaptive Sampling

This chapter presents our new probabilistic path planner, called *Workspace-based Connectivity Oracle* (WCO). Unlike WIS that generates static sampling distribution, WCO generates *dynamic sampling distribution*. WCO combines workspace information with sampling history and the current state of the roadmap to dynamically adapt its sampling distribution over time. Furthermore, unlike WIS that represents the robot using a feature point, WCO uses *multiple feature points*, such that more robot's information can be incorporated to the planner. Each feature point composes a component sampler that uses workspace information to estimate regions of the robot's free-space $\mathcal{F}$ that is more likely to improve the quality of the current roadmap, and then sample more densely in these regions. So, WCO is composed of many component samplers. These samplers are combined using the adaptive hybrid sampling approach which is based on the samplers's sampling histories. Our analysis shows that the failure probability of WCO converges to zero exponentially in the number of milestones, whenever a solution exists. And when WCO's estimation is good, the upper bound on the failure probability of WCO is lower than that of the basic-PRM. Our experimental results show that WCO performs up to 28 times faster than the basic-PRM. And as the dimensionality of $\mathcal{C}$ increases, WCO's performance decreases slower than other recent probabilistic path planners. In the attached demo, we show that WCO is able to solve a bridge

inspection problem involving a 35 dofs robot.

We start by presenting the main idea of WCO in Section 4.1. Unlike WIS that uses geometric property of the workspace free-space $\mathcal{W}_{\mathcal{F}}$ to guide sampling, the main idea of WCO is to use paths in $\mathcal{W}_{\mathcal{F}}$ to guide sampling. We explore the relation between paths in $\mathcal{F}$ and in $\mathcal{W}_{\mathcal{F}}$ in Section 4.2. An overall strategy of the WCO planner is then presented in Section 4.3. The details of the planner are presented in Section 4.4 and Section 4.5. Next, implementation details and experimental results are presented in Section 4.6. While an analysis of WCO is presented in Section 4.7. Finally, we end with a discussion of possible improvements of the planner in Section 4.8.

4.1 The Idea

In Chapter 2, we have argued that adaptive sampling is a promising way of speeding-up probabilistic path planning. It incrementally infers partial knowledge of key geometric properties of $\mathcal{F}$ during roadmap construction and uses this knowledge to dynamically adapt the sampling distribution. Considering that a priori construction of $\mathcal{F}$ is infeasible and a suitable sampling distribution is dependent on the geometry of $\mathcal{F}$, adaptive sampling provides a way to generate a more suitable sampling distribution without a priori construction of $\mathcal{F}$. Furthermore, adaptive sampling reduces oversampling regions of $\mathcal{F}$ that have been well-represented by the current roadmap.

However, to infer geometric properties of $\mathcal{F}$, existing probabilistic planners with adaptive sampling [Kavraki et al., 1996b, Morales et al., 2004, Burns and Brock, 2005, Hsu et al., 2005, Rodriguez et al., 2006] use only sampling history. This is inadequate because to learn the usefulness of sampling a particular region of $\mathcal{F}$, the planner needs many samples in or around the region. This is difficult to achieve in narrow passages, which are often crucial for capturing the connectivity of $\mathcal{F}$.

To address this issue, WCO uses both workspace information and sampling history. WCO is based on the standard multi-query PRM approach as described in Section 2.2. Since there is no confusion, in the rest of this thesis, we use WCO

to refer to both the sampling strategy and the planner.

WCO is an ensemble sampler composed of many component samplers. Each sampler is based on a feature point of the robot. They are based on a key observation: a collision-free path between a start configuration q_i and a goal configuration q_g in a robot's free-space $\mathcal{F}$ implies a collision-free path in the workspace free-space $\mathcal{W}_\mathcal{F}$ for *every* points in the robot between the corresponding start and goal positions of the point. So, if we find a collision-free path in $\mathcal{W}_\mathcal{F}$ for every point in the robot and all these paths correspond to the same path ψ in $\mathcal{F}$, then ψ is indeed a collision-free path in $\mathcal{F}$ for the robot to move from q_i to q_g. Finding a path for every point is, of course, impractical. Nevertheless, we can use a set of points, called feature points, on the robot to predict regions of $\mathcal{C}$ that are more likely to be useful for connecting disconnected components of a roadmap. So, each WCO component sampler is based on a feature point of the robot. They are then combined, based on their sampling histories, using the adaptive hybrid sampling (AHS) approach [Hsu et al., 2005], which is a restricted form of reinforcement learning.

4.2 Paths in $\mathcal{F}$ and in $\mathcal{W}_\mathcal{F}$

Unlike WIS that uses geometric property of $\mathcal{W}_\mathcal{F}$ to guide sampling, WCO uses paths in $\mathcal{W}_\mathcal{F}$ to guide its sampling. In this section, we try to understand the relation between paths in $\mathcal{F}$ and in $\mathcal{W}_\mathcal{F}$.

Observation on the relation between paths in $\mathcal{F}$ and in $\mathcal{W}_\mathcal{F}$ gives us the following proposition, which is the key observation utilized by WCO.

Proposition 4.1 *If two configurations $q, q' \in \mathcal{C}$ are connected by a path in $\mathcal{F}$, then for any point a in a robot, $P_a(q)$ and $P_a(q')$, the projections of q and q' in $\mathcal{W}_\mathcal{F}$, are connected by a path in $\mathcal{W}_\mathcal{F}$.*

To understand the relation between paths in $\mathcal{F}$ and in $\mathcal{W}_\mathcal{F}$ further, we explore the relation between the possible length of paths in $\mathcal{F}$ and the possible length of paths in $\mathcal{W}_\mathcal{F}$. To state the relation between the possible length of paths in $\mathcal{F}$ and paths in $\mathcal{W}_\mathcal{F}$, we need additional notation. We define a linking sequence $\mathcal{L}(q, q')$

between two configurations $q, q' \in \mathcal{F}$ as a sequence of configurations $q^1, \ldots, q^n \in \mathcal{F}$ such that each configuration in the sequence lies in the visibility set of its previous configuration, *i.e.*, $\bigwedge_{i=2}^{n} q^i \in \mathcal{V}(q^{i-1})$ where $q^1 = q$, $q^n = q'$, and $\mathcal{V}(\cdot)$ is the visibility set. We define the length $|\mathcal{L}(q, q')|$ of a linking sequence $\mathcal{L}(q, q')$ as the number of configurations in $\mathcal{L}(q, q')$. To articulate the relation between $\mathcal{F}$ and $\mathcal{W}_{\mathcal{F}}$, we also use the same denifition to define linking sequence in $\mathcal{W}_{\mathcal{F}}$.

Now, for any robot and any point a on the robot such that the projection P_a is linear, we can state the relation between paths in $\mathcal{F}$ and in $\mathcal{W}_{\mathcal{F}}$ as

Theorem 4.1 *Let's denote A as the set of points a on the robot such that the projection P_a is linear. For any $a \in A$, if $X \subseteq \mathcal{W}_{\mathcal{F}}$ is a path-connected subset that encloses $P_a(q)$ and $P_a(q')$, then any linking sequence $\mathcal{L}(q, q')$ in $L_a(X)$ is at least as long as the shortest linking sequence between $P_a(q)$ and $P_a(q')$ in X.*

Proof We prove by contradiction. Let's denote the length of the shortest linking sequence between $P_a(q)$ and $P_a(q')$ in X as n. Suppose there is a linking sequence $\mathcal{L}(q, q') = q^1, q^2, \ldots, q^k$ in $L_a(X)$ where $k < n$, $q^1 = q$, and $q^k = q'$. Since $\mathcal{V}(q) \subseteq L_a(\mathcal{V}(P_a(q)))$ (Theorem 3.1) and since a linking sequence means $q^i \in \mathcal{V}(q^{i-1})$, $\forall_{i\in[2,k]}\ P_a(q^i) \in \mathcal{V}(P_a(q^{i-1}))$. Since $\mathcal{L}(q, q')$ lies entirely in $L_a(X)$, $\forall_{i\in[1,k]}\ P_a(q^i) \in X$. This means, there is a linking sequence of length $k < n$ between $P_a(q)$ and $P_a(q')$ in X. But this contradicts our assumption that the shortest linking sequence between $P_a(q)$ and $P_a(q')$ in X is n. $\square$.

The main difficulties in relaxing the above theorem such that it holds for any point a on the robot and any types of robot is that $\mathcal{V}(q) \subseteq L_a(\mathcal{V}(P_a(q)))$ does not always hold for arbitrary points in the robot and for arbitrary types of robot. However, we can relax the above theorem by restricting our definition of linking sequence by the distance between a configuration to the forbidden region. The distance function is as defined in Section 3.2. Let's denote our restricted linking sequence between two configurations $q, q' \in \mathcal{F}$ as $\mathcal{L}_r(q, q')$, and define it as a sequence of configurations $q^1, \ldots, q^n \in \mathcal{F}$ such that $\bigwedge_{i=2}^{n} (q^i \in \mathcal{B}(q^{i-1}, \Delta_{\mathcal{O}}(q^{i-1})) \cap \mathcal{V}(q^{i-1}))$ where $q^1 = q$, $q^n = q'$, and $\mathcal{B}(q, \Delta_{\mathcal{O}}(q))$ is the set of configurations within $\Delta_{\mathcal{O}}(q)$ distance from q. Notice that for any collision-free configuration q, the intersection $\mathcal{B}(q, \Delta_{\mathcal{O}}(q)) \cap \mathcal{V}(q)$ is $\mathcal{B}(q, \Delta_{\mathcal{O}}(q))$. From the definition of distance between

two configurations in Section 3.2, any configuration within $\Delta_{\mathcal{O}}(q)$ from q must be collision-free, which means any configuration in $\mathcal{B}(q, \Delta_{\mathcal{O}}(q))$ can see any other configuration in $\mathcal{B}(q, \Delta_{\mathcal{O}}(q))$. Hence, the requirement of our restricted linking sequence can be simplified to a sequence of configurations $q^1, \ldots, q^n \in \mathcal{F}$ such that $\bigwedge_{i=2}^{n} q^i \in \mathcal{B}(q^{i-1}, \Delta_{\mathcal{O}}(q^{i-1}))$. As in the definition of linking sequence, we also use the same definition to define restricted linking sequence in $\mathcal{W}_{\mathcal{F}}$. We can then relax Theorem 4.1 to:

Theorem 4.2 *Let $\mathcal{A}$ be a robot and a be any point in $\mathcal{A}$. If $X \subseteq \mathcal{W}_{\mathcal{F}}$ is a path-connected subset that encloses $P_a(q)$ and $P_a(q')$, then any restricted linking sequence $\mathcal{L}_r(q, q')$ in $L_a(X)$ is at least as long as the shortest restricted linking sequence between $P_a(q)$ and $P_a(q')$ in X.*

Proof First, we show that for any configuration $q \in \mathcal{F}$, $\mathcal{B}(q, \Delta_{\mathcal{O}}(q)) \subseteq L_a(\mathcal{B}(P_a(q), \delta_{\mathcal{O}}(P_a(q))))$. By contradiction, suppose there is a configuration $q' \in \mathcal{B}(q, \Delta_{\mathcal{O}}(q))$ where $P_a(q') \notin \mathcal{B}(P_a(q), \delta_{\mathcal{O}}(P_a(q)))$. From the definition of distance between two configurations in Section 3.2, for q' to be in $\mathcal{B}(q, \Delta_{\mathcal{O}}(q))$, the distance between $P_a(q')$ and $P_a(q)$ is at most $\min_{a \in \mathcal{A}} \delta_{\mathcal{O}}(P_a(q))$. However, to satisfy $P_a(q') \notin \mathcal{B}(P_a(q), \delta_{\mathcal{O}}(P_a(q)))$, the distance between $P_a(q')$ and $P_a(q)$ must be more than $\delta_{\mathcal{O}}(P_a(q))$. This is a contradiction, and hence for any configuration $q \in \mathcal{F}$, $\mathcal{B}(q, \Delta_{\mathcal{O}}(q)) \subseteq L_a(\mathcal{B}(P_a(q), \delta_{\mathcal{O}}(P_a(q))))$.

It is then straightforward to adopt the proof in Theorem 4.1 to prove this theorem. By contradiction, suppose n is the shortest restricted linking sequence between $P_a(q)$ and $P_a(q')$ in X and suppose there is a restricted linking sequence between two configurations $q, q' \in \mathcal{F}$, $\mathcal{L}_r(q, q') = q^1, \ldots, q^k$ in $L_a(X)$ where $k < n$, $q^1 = q$, and $q^k = q'$. Since $\mathcal{B}(q, \Delta_{\mathcal{O}}(q)) \subseteq L_a(\mathcal{B}(P_a(q), \delta_{\mathcal{O}}(P_a(q))))$ and by our definition of a restricted linking sequence, $\forall_{i \in [1,k]}\ P_a(q^i) \in \mathcal{B}(P_a(q^{i-1}), \Delta_{\mathcal{O}}(P_a(q^{i-1}))))$. Since the restricted linking sequence is enclosed in $L_a(X)$, there is a restricted linking sequence in X between $P_a(q)$ and $P_a(q')$ with length $k < n$. This contradicts our assumption that n is the shortest restricted linking sequence between $P_a(q)$ and $P_a(q')$ in X. $\square$.

In the rest of the discussion, we will use linking sequence to refer to both $\mathcal{L}$ and $\mathcal{L}_r$. The above theorems imply that to find a path that connects two disconnected

components of a roadmap, reducing our search space to subsets of $\mathcal{W}_{\mathcal{F}}$ that contains short linking sequences is preferable than to subset that contains only long linking sequences. In a roadmap, a path between two collision-free configurations q and q' are made up of a sequence of straight-line segments, where their end-points are either the query configurations or the milestones in the roadmap. The sequence of milestones that connects q and q', inclusive of q and q', can be considered as a linking sequence between q and q'. To quickly connect two disconnected components of a roadmap, we want to find a sequence of milestones that construct short linking sequence between configurations from different roadmap components. Theorem 4.2 implies that for any types of robots and for any points in the robot, it is less likely that we find short linking sequence between q and q' if the search is restricted to $L_a(X)$ where $X \subseteq \mathcal{W}_{\mathcal{F}}$ contains only long linking sequences between $P_a(q)$ and $P_a(q')$. Of course there are cases where the shortest linking sequence in $L_a(X')$, where $X' \subseteq \mathcal{W}_{\mathcal{F}}$ contains significantly shorter linking sequences than X, is significantly longer than the shortest linking sequence in $L_a(X)$. Or even worse, $L_a(X')$ may not contain any linking sequence between q and q'. But, as we will see in the experimental results (Section 4.6) and analysis (Section 4.7), choosing short linking sequence is effective.

Note that one may combine the size of the linking sequence in $X \subseteq \mathcal{W}_{\mathcal{F}}$ with the size of the visibility sets of points in X, to get a better estimation of the difficulty of finding a path by restricting sampling to $L_a(X)$. However, for simplicity, WCO uses only the small linking sequence preference.

4.3 Overview of WCO

Now, we give an overall strategy of our new planner, Workspace-based Connectivity Oracle (WCO). During roadmap construction, WCO maintains a partially constructed roadmap $\mathcal{R}$. Distinct connected components of $\mathcal{R}$ may in fact lie in the same connected component of $\mathcal{F}$, due to inadequate sampling of certain critical regions. To sample such regions, WCO examines the workspace paths of a set of *feature points* in the robot and constructs a sampler for each feature point a. To

connect two components $\mathcal{R}_1$ and $\mathcal{R}_2$ of $\mathcal{R}$, we use P_a to project the milestones of $\mathcal{R}$ into $\mathcal{W}_\mathcal{F}$ and search for "channels" in $\mathcal{W}_\mathcal{F}$ that connect the projected milestones of $\mathcal{R}_1$ and $\mathcal{R}_2$. In trying to keep the number of milestones low, we prefer to use short channels instead of longer ones (Theorem 4.2). Furthermore, as we will see in Section 4.7.2, short channels tend to improve the performance of WCO. The channels suggest the regions of $\mathcal{C}$ that is more likely to connect $\mathcal{R}_1$ and $\mathcal{R}_2$. So, we use L_a to lift the channels into $\mathcal{C}$ and adapt the distribution to sample more densely in the regions covered by the lifted channels. To be sensitive to the changes in $\mathcal{R}$, WCO adapts its sampling distribution incrementally whenever a new milestone is added to $\mathcal{R}$.

Although workspace-based PRM planners often consider only a single feature point [Foskey et al., 2001,van den Berg and Overmars, 2005], this is inadequate. By Proposition 4.1, a collision-free path in $\mathcal{C}$ implies a collision-free path in $\mathcal{W}$ for every point in the robot. So, we use a set of pre-selected feature points and construct an independent sampler s_i for each feature point a_i, $i = 1, 2, \ldots, L-1$. We make two simplifying assumptions. First, a finite number of feature points are sufficient to indicate the important regions of $\mathcal{C}$ for sampling. Second, we can treat the feature points independently. These two assumptions reduce the computational cost and are shown to be effective in identifying important regions of $\mathcal{C}$ (see Section 4.6). Despite the independence assumption, the kinematic constraints of a robot are not entirely ignored. Implicitly, WCO assigns higher sampling density to regions obeying such constraints. We will discuss more about this in Section 4.5.2. To provide roadmap coverage, we add a uniform component sampler s_0 to the WCO component samplers $s_1, s_2, \ldots, s_{L-1}$ and form the set $\mathcal{S} = \{s_0, s_1, s_2, \ldots, s_{L-1}\}$. The component samplers in $\mathcal{S}$ are combined through the AHS [Hsu et al., 2005] approach to form an ensemble sampler. With this approach, each component sampler has an associated weight proportional to the probability of it being used, and the weights are adjusted to reflect the success of the sampler according to the sampling history.

The overall strategy of WCO can be seen in Algorithm 4.1. It is based on the standard multi-query PRM approach as described in Section 2.2. It accepts a set

Algorithm 4.1 WCO(Q, N)

1: Extract workspace information.
2: Let $p_{S_j}(s_i)$ be the probability of picking a component sampler s_i at iteration-j. Initialize $p_{S_0}(s_i) = 1/L$ for $i = 0, 1, \ldots, L-1$.
3: Set the current number of milestones $j = 0$.
4: **repeat**
5: Pick a component sampler s_i from $\mathcal{S} = \{s_0, \ldots, s_{L-1}\}$ with probability p_{S_j}.
6: Run s_i to sample a new configuration q.
7: **if** `FreeConf`(q) is true **then**
8: Insert q to the roadmap $\mathcal{R}$.
9: Update the distributions of $s_i, i = 1, 2, \ldots, L-1$.
10: Compute the probabilities $p_{S_{j+1}}(s_i), i = 0, \ldots, L-1$.
11: Increment j by 1.
12: **until** `AllSolved`(Q, $\mathcal{R}$, Ψ) is true or $\mathcal{R}$ contains N milestones.
13: Return Ψ.

of queries Q and maximum number of milestones N as inputs. And returns a set of paths Ψ between each query in Q. The details of the primitive `FreeConf` for checking whether a configuration is collision free or not and the primitive `AllSolved`() for checking whether all queries have been solved or not are presented in Appendix A. The details for a component sampler (Algorithm 4.1 line 1 and 6–9) are presented in Section 4.4 while the details of combining the samplers (Algorithm 4.1 line 2 and 10) are presented in Section 4.5.

4.4 Details of a Single Component Sampler

We now describe the construction of a component sampler of WCO for a fixed feature point a, specifically, how to extract workspace connectivity (Algorithm 4.1 line 1) in Section 4.4.1, how to adapt the sampling distribution (Algorithm 4.1 line 9) in Section 4.4.2, and how to take a sample for rigid and articulated robots (Algorithm 4.1 line 6) in Section 4.4.3.

4.4.1 Extracting Workspace Connectivity

Just as WIS (Chapter 3), WCO starts by computing a cell decomposition of $\mathcal{W}_{\mathcal{F}}$. However unlike WIS, any cell decomposition method, *e.g.*, triangulation, quadtrees, regular grids, etc., can be used for WCO, because the purpose of cell decomposition

in WCO is to extract the connectivity information of $\mathcal{W}_{\mathcal{F}}$ and to discretize the sampling space. After $\mathcal{W}_{\mathcal{F}}$ is decomposed, the adjacency graph of the cells is computed.

Building on our work on WIS, WCO uses triangulation. We will use the term triangulation in a general sense, *i.e.*, referring to triangulation for 2D workspace and tetrahedralization for 3D workspace. WCO samples the boundary of obstacles in $\mathcal{W}_{\mathcal{F}}$ and constructs a Delaunay triangulation [de Berg et al., 2000] over the sampled points. Under reasonable geometric assumptions, the constructed triangulation is conforming [Amenta et al., 2001], meaning that every triangle in the resulting triangulation lies either entirely in $\mathcal{W}_{\mathcal{F}}$ or its complement. Although helpful, this property is not required for our purposes. WCO will then consider the triangles that lie entirely or partially in $\mathcal{W}_{\mathcal{F}}$ as the triangles in $\mathcal{W}_{\mathcal{F}}$. Let's denote these triangles as $\mathcal{T}_{\mathcal{F}}$. The adjacency graph G is then defined as the graph whose vertices represent the triangles in $\mathcal{T}_{\mathcal{F}}$ and two vertices v_1 and v_2 are connected by an edge in G whenever the two triangles represented by v_1 and v_2 shares a boundary.

4.4.2 Adapting Sampling Distribution

A skeleton of a WCO component sampler is shown in Algorithm 4.2. Let us now look at how it represents and updates the sampling distribution based on workspace channels $\mathcal{T}'$. During the roadmap construction, WCO maintains a partially constructed roadmap $\mathcal{R}$. To sample a new milestone, each component sampler maintains a separate sampling distribution p_{T_j} defined over $\mathcal{T}_{\mathcal{F}}$, where T_j is the random variable that represents the sampled triangle at iteration j. The distribution p_{T_j} assigns equal probabilities to all triangles in the workspace channels $\mathcal{T}'$ and zero probabilities to all other triangles in $\mathcal{T}_{\mathcal{F}}$.

Since workspace channels estimate regions of $\mathcal{W}_{\mathcal{F}}$ that is more likely to be passed by the robot in order to connect disconnected components of the roadmap, to find workspace channels, we first need to project the connected component information of the roadmap to the workspace. For this, we start by projecting milestones of $\mathcal{R}$ to $\mathcal{W}_{\mathcal{F}}$ (Algorithm 4.2, line 4). Suppose that a milestone m

Algorithm 4.2 A WCO component sampler.

1: Given a feature point a, sample a configuration q, based on the sampling distribution defined over the decomposition $\mathcal{T}_{\mathcal{F}}$.
2: **if** `FreeConf`(q) is true **then**
3: Insert q to the roadmap $\mathcal{R}$.
4: Project q to $\mathcal{W}$ and let $t \in \mathcal{T}_{\mathcal{F}}$ be the triangle that contains $P_a(q)$.
5: Update the label sets and the end-points sets of all affected triangles in $\mathcal{T}_{\mathcal{F}}$.
6: Insert terminal t as a vertex of the channel graph G', if it is not yet in G'.
7: Perform a breadth-first search in the adjacency graph G from t until reaching the first terminal t' other than t.
8: **if** t and t' hold different label sets **then**
9: Add the path between t and t' to G'.
10: Delete (parts of) the paths that are unlikely to connect different components of the current roadmap.
11: Update the workspace channels $\mathcal{T}'$.
12: Update the sampling distribution.

belongs to a roadmap component $\mathcal{R}_i$ of $\mathcal{R}$. We associate $\mathcal{R}_i$ with the triangle $t \in \mathcal{T}_{\mathcal{F}}$ that contains $P_a(m)$. Thus, each triangle t contains a set of labels that indicates the roadmap components which t is associated with. We call a triangle t a *terminal* whenever its label set is non-empty, meaning that t contains at least one projected milestone. See Figure 4.1*a* for an example.

Next, we find channels that connect terminals with different label sets by considering the adjacency graph G. We compute a subgraph of G, called a *channel graph* G', that spans all the terminals and connect them together. As argued in Section 4.2, we prefer to find short channels. In Section 4.7, we will discuss more on why small workspace channels are more desirable. The intuition behind the channel graph is very much like that of a roadmap in the configuration space: it uses simple paths, in this case, the shortest paths to connect every pair of two terminals that are close to each other and have different label sets. The workspace channels $\mathcal{T}'$ are then the triangles corresponding to these non-isolated vertices of G'. See Figure 4.1 for an example.

The channel graph is computed incrementally (Algorithm 4.2, lines 6–9), as new milestones are added. The incremental construction allows WCO to respond to changes in $\mathcal{R}$ and simplifies computation. To see that G' indeed "connects" all the terminals together, note that the channel graph G' clearly contains all the terminals. Furthermore, it is *weakly connected* in the sense that between every pair

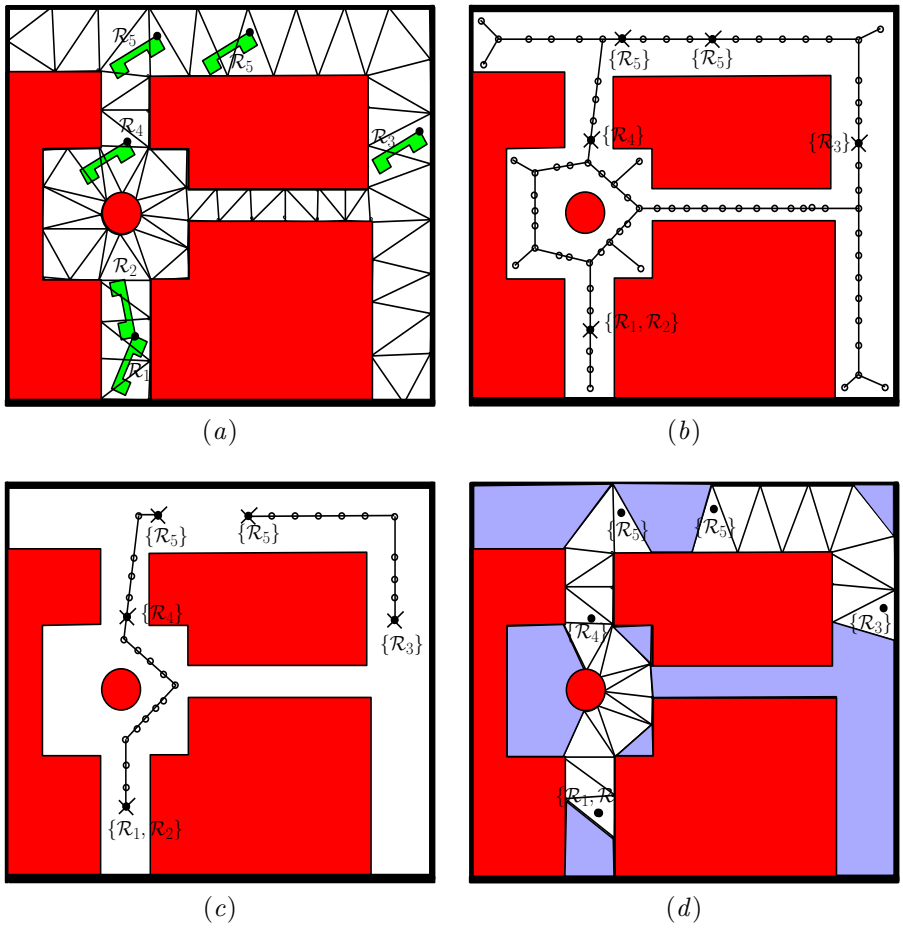


Figure 4.1: Illustration of WCO sampling strategy. Obstacles are colored red. The robot is a free-flying rigid body robot and is green colored. (a) Milestones are projected to the triangulated workspace. The labels indicate the roadmap components to which the milestones belong. The feature point of the rigid robot is marked by a black dot. (b) The adjacency graph G. Terminals are marked by crosses. (c) The channel graph G' . Paths that connect terminals with the same label set (*e.g.*, the path between the two terminals labelled $\{\mathcal{R}_5\}$) are not in G', as they connect terminals corresponding to milestones in the same connected components of $\mathcal{R}$ and hence unlikely to help in improving the connectivity of $\mathcal{R}$. (d) The workspace channels $\mathcal{T}'$ is the white-colored triangles.

of terminals t and t', there is a sequence of terminals $t_i, i = 1, 2, \ldots, n$ with $t_1 = t$ and $t_n = t'$ such that every adjacent pair t_i and t_{i+1} either have exactly the same label set or have a path between them in G'. In the example shown in Figure 4.1*c*, the two terminals $\{\mathcal{R}_3\}$ and $\{\mathcal{R}_4\}$ are weakly connected.

In trying to keep the size of the workspace channels small, we delete (parts of) the paths that now connect terminals with the same label sets (Algorithm 4.2, lines 10). To quickly identify which vertices of G' to be deleted, each vertex

keeps an *end-points set*. An end-points set $E(v)$ of a vertex v is a set of 3-tuples $< ls_1, ls_2, ts >$ that records which pairs of labels sets are connected through the workspace path(s) that passes through the vertex v. The components ls_1 and ls_2 are the label set of the terminals connected by a path passing through v, while ts is a time-stamp that indicates the most recent time among the time when the 3-tuple is inserted and the time ls_1 or ls_2 is updated. For example, suppose at iteration-j, a path passing trough vertex v and connecting t to t' is inserted to G'. Then, a 3-tuple of <label set of t, label set of $t', j >$ is inserted to $E(v)$.

The question remains as to when a vertex v should be deleted from G'. Recall that the label set of a terminal indicates the roadmap components associated with the terminal. So, when ls_1 and ls_2 of a 3-tuple in $E(v)$ of a vertex v are the same, v does not seem useful in improving the connectivity of the current roadmap and hence deleteing v from G' seems reasonable. However, $E(v)$ may contain more than one 3-tuples and the other 3-tuples of $E(v)$ may have different value of ls_1 and ls_2. This indicates that the vertex v may still be useful in connecting different components of the roadmap and deleting v may not be desirable. Ideally, we will only delete a vertex v from the channel graph if each element of $E(v)$ has the same ls_1 and ls_2 (*i.e.*, for each element of $E(v)$, $ls_1 = ls_2$). However, this means our workspace channels will often be quite large because often times a vertex is identified as potentially useful for connecting many pairs of different terminals. To keep the size of the workspace channels small, we use the time-stamp as a heuristic to decide whether a vertex should be deleted or not. With this heuristic, a vertex v is deleted whenever each of the k most recent elements of $E(v)$ have the same ls_1 and ls_2, where k is a constant.

The above heuristic generates a spectrum, where the smallest workspace channels is generated when we set $k = 1$. And the largest workspace channels is generated when k is unbounded, such that a vertex is deleted if each element of its end-points set has the same ls_1 and ls_2. Although the weakly connected property is satisfied only when k is unbounded, in practice the workspace channels generated when k is unbounded becomes too large such that the performance of WCO suffers. Discussion on the effect of the size of workspace channels to the overall

performance of WCO is presented in Section 4.7.2. On the other hand, considering that we update the workspace channels everytime a new milestone is inserted, we can expect that problems that may occur because the workspace channels are not weakly connected can be reduced by the fast update. Therefore, we would prefer to use the smallest channel possible rather than keeping the weakly connected property at the cost of enlarging the size of the channels. And it turns out that in our experiments, the results when $k = 1$ are quite well. So, we will delete a vertex $E(v)$ whenever the element of $E(v)$ with the most recent time-stamp has the same ls_1 and ls_2.

4.4.3 Sampling a Configuration

As WIS, WCO samples a configuration from $\mathcal{C}$ (Algorithm 4.2, line 1) in two stages. First, at iteration j, WCO samples a point $\mathbf{x} \in \mathcal{W}_{\mathcal{F}}$ by picking a triangle $t \in \mathcal{T}_{\mathcal{F}}$ according to the distribution p_{T_j} and then picking a point $\mathbf{x} \in t$ uniformly at random. Next, it samples a configuration from $L_a(\mathbf{x})$. Note that to ease finding channels, the uniform component sampler uses the same two-stage sampling procedure. However, it samples a point $\mathbf{x}$ from $\mathcal{W}_{\mathcal{F}}$ uniformly at random. The details of sampling a configuration from $L_a(\mathbf{x})$ depends on the robot's kinematics and are the same as in WIS. For convenience, below we summarize the strategies for rigid and articulated robots.

The configuration q of a rigid robot consists of a positional component q_τ, which specifies the position of the robot's reference point in the workspace, and an orientational component q_θ, which specifies the orientation of the robot. To sample a configuration, WCO first picks q_θ uniformly at random. It then picks a point $\mathbf{x} \in \mathcal{W}_{\mathcal{F}}$, as described above, and compute q_τ so that at $q = (q_\tau, q_\theta)$, the robot's feature point a lies at $\mathbf{x}$ and the robot has orientation q_θ.

For an articulated robot, the configuration q specifies its joints parameters $q_1, q_2, \ldots$. Suppose that the feature point a lies in the ℓth link of the robot. To sample a configuration, WCO again picks a point $\mathbf{x} \in \mathcal{W}_{\mathcal{F}}$ and then find the joint parameters $q_1, q_2, \ldots, q_\ell$ that place a at $\mathbf{x}$ by solving the robot's inverse kinematics (IK) equations. If IK has no solution, WCO picks another $\mathbf{x}$. If IK has more

than one solution, WCO picks one at random. WCO then samples the other joint parameters $q_{\ell+1}, q_{\ell+2}, \ldots$ uniformly at random. Various improvements can be made to speed-up this process. For instance, the sampling domain may be restricted according to the reachability of each feature point.

4.5 Constructing the Ensemble Sampler

Now, we describe how WCO combines the component samplers. We start by presenting the way WCO combines its component samplers in Section 4.5.1. And then continue by presenting how multiple feature points helps WCO planner in Section 4.5.2. Finally, we give a heuristic for choosing the feature points in Section 4.5.3.

4.5.1 Combining Samplers through AHS

Recall from Section 4.3 that WCO uses a set of component samplers, $\mathcal{S} = \{s_0, s_1, \ldots, s_{L-1}\}$, where s_0 is a uniform component sampler and each $s_i, i = 1, 2, \ldots, L-1$ is based on a feature point of the robot. Note however that for efficiency of updating the sampling distribution of each WCO component sampler, *i.e.*, to help project the sampled milestone to $\mathcal{W}_{\mathcal{F}}$, the uniform component sampler performs uniform sampling over $\mathcal{W}_{\mathcal{F}}$ and not over $\mathcal{F}$. It follows the two-stage sampling presented in Section 4.4.3, but the distribution defined over $\mathcal{W}_{\mathcal{F}}$ is uniform distribution.

We combine the component samplers through AHS. Each sampler s_i has an associated weight w_i, which reflects the usefulness of s_i according to its sampling history. The sampler s_i is chosen to run with probability $p_{S_j}(s_i)$ that depends on the weight w_i at time j. To adapt the ensemble distribution (Algorithm 4.1, line 10), WCO adjusts the weights so that the component samplers with better performance have higher weights.

In iteration j of Algorithm 4.1, WCO chooses s_i with probability

$$p_{S_j}(s_i) = (1-\gamma)\frac{w_i(j)}{\sum_{k=0}^{L-1} w_k(j)} + \frac{\gamma}{L}, \tag{4.1}$$

where S_j is the random variable that represents the sampler picked at time j, $w_i(j)$ is the weight of s_i in iteration j and $\gamma \in (0, 1]$ is a small fixed constant. WCO uses the chosen s_i to sample a new milestone m and assigns to s_i a reward r that depends on the effect of m on the roadmap $\mathcal{R}$:

- The milestone m reduces the number of connected components of $\mathcal{R}$. In this case, m merges two or more connected components and improves its connectivity. We set $r = 1$.
- The milestone m increases the number of connected components of $\mathcal{R}$. In this case, m creates a new connected component and potentially improves the coverage of $\mathcal{R}$. We also set $r = 1$.
- Otherwise, $r = 0$.

We then update the weight of s_i:

$$w_i(j+1) = w_i(j) \exp\left((r/p_{S_j}(s_i))\gamma/L\right). \tag{4.2}$$

where $w_i(0) = 1$ for $i \in [0, L-1]$. Note that the exponent depends on the received reward r weighted by the probability $p_{S_j}(s_i)$ of choosing s_i at time j. It makes the weight of a component sampler increases more when $p_{S_j}(s_i)$ is low and less when $p_{S_j}(s_i)$ is high. As a result, the planner is responsive to the changes in the performance of a sampler, but mere luck will not cause a sampler to be favored much. If a sampler is not chosen, then its weight remains the same as before, *i.e.*, $w_i(j+1) = w_i(j)$.

Note that here we slightly modify the AHS presented in [Hsu et al., 2005]. We can choose a different component sampler after a configuration is sampled, regardless of whether the configuration is collision-free or not. By doing so, the planner avoids getting stuck at one of the component sampler that may require a huge amount of samples before a collision-free configuration is sampled. Furthermore, since the cost used by different component samplers in WCO are almost the same, we do not incorporate cost in our weighting. More details on AHS are available in [Hsu et al., 2005] and further exploration on the behavior of this method is presented in the next chapter.

Although there are many possible schemes for updating the weights, AHS has an important advantage. It can be shown that under suitable assumptions, the ensemble sampler generated by AHS is competitive against the best component sampler [Hsu et al., 2005]. More precisely, the following competitive ratio holds:

$$R_{\max} - R \leq (\mathrm{e} - 1)\gamma R_{\max} + \frac{L \ln L}{\gamma}, \tag{4.3}$$

where R is the expected total reward received by the ensemble sampler and $R_{\max}$ is the total reward received by the best component sampler if it is always chosen to run. This result can be interpreted as saying that the ensemble sampler performs almost as well as the best component sampler, without knowing in advance which component sampler is the best. With some small variations on the scheme for updating the weights, one can also show that the modified ensemble sampler is competitive against any linearly weighted combination of component samplers, an even stronger result theoretically [Auer et al., 2002]. This guaranteed performance is one reason why we choose AHS for combining component samplers.

4.5.2 Why Multiple Feature Points ?

Although in Chapter 3, we have discussed that representing the robot with a single feature point seems insufficient, it is not clear how exactly using multiple feature points helps the WCO sampling strategy. There are two main advantages of using multiple feature points.

First, the use of multiple feature points relaxes the restriction of choosing the "best" feature point, which is difficult to do. For WCO with single feature point to perform well, the feature point must generate a suitable sampling distribution over the entire $\mathcal{C}$. In general, such a sampler is difficult to construct. Using multiple feature points simplifies the task. It is sufficient for a component sampler to work well in only part of $\mathcal{C}$, provided that several component samplers can be combined effectively to generate a suitable distribution over the entire $\mathcal{C}$.

Now, when all feature points generate the same workspace channels, *e.g.*, the example in Figure 4.2, one may doubt the use of multiple feature points. When

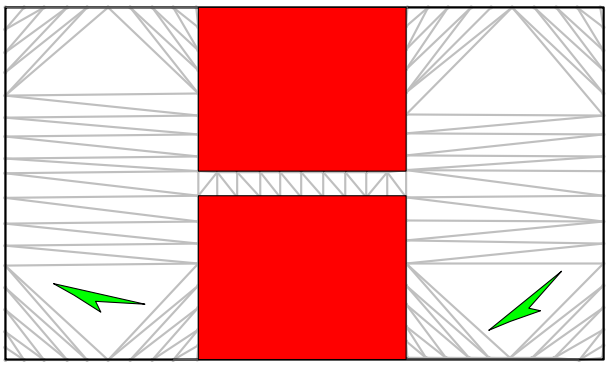


Figure 4.2: Illustration of multiple copies of the same workspace channels. The obstacles are the red-colored rectangles. The robot is colored green. The initial configuration is shown by the robot at the left part of the scene, while the goal configuration is shown by the robot at the right part of the scene. Assuming that the current roadmap consists of only the initial and goal configurations, each point in the robot will generate exactly the same workspace channels.

different feature points traverse different paths and generate different workspace channels, it is straightforward to see that in general, different feature points generate different sampling distributions over $\mathcal{C}$. Some distributions may be more suitable for sampling different regions of $\mathcal{F}$. Which distribution is more suitable for sampling which regions of $\mathcal{F}$ can then be inferred from the sampling history. Nevertheless, in many planning scenarios, it is quite likely that different feature points traverse similar path and generate the same workspace channels. One may then wonder, how does having multiple copies of the same workspace channels help in generating a more suitable sampling distribution? However, a WCO component sampler samples a configuration from the lift mapping of the workspace channels. Since given different feature points, the lift mappings of the same workspace channels are different, different feature points may generate different sampling distribution over $\mathcal{C}$. So in general, different feature points generate different sampling distribution over $\mathcal{C}$, even though the workspace channels they generate may be the same.

The second advantage of using multiple feature points is that it implicitly takes kinematic constraints into consideration. WCO simplifies computation by assuming that each feature point of the robot is independent. However, the way WCO combines the component samplers implicitly takes kinematic constraints of

the feature points into account. Without loss of generality, suppose we use two WCO component samplers s_1 and s_2. Points in the workspace channels of s_1 and points in the workspace channels of s_2 correspond to the same set of configurations Q when the kinematic constraints between s_1 and s_2 in the robot are satisfied. The probability of sampling a configuration in Q is then a combination of the probability that s_1 samples Q and the probability that s_2 samples Q. This means in general, the probability of sampling Q is higher than the probability of sampling other configurations that only correspond to the workspace channels of s_1 or s_2 alone. So, points inside the workspace channels where the kinematics constraints of the feature points are satisfied correspond to configurations with higher sampling density. This means, despite the independent assumption, WCO with multiple feature points implicitly takes kinematic constraints into consideration.

4.5.3 Choosing the Feature Points

So far, we have assumed that the feature points have been determined, but at some point we do need to choose the feature points. The idea is to choose a small set of representative points that capture the geometry of the robot sufficiently. We would like the set of feature points to be small because we construct a component sampler for each feature point. A large number of component samplers increase both the difficulty of identifying the good ones through AHS and the computational cost for updating the sampling distribution.

To choose such set of feature points, we use a heuristic that chooses points that are spaced far apart. The reason is that feature points close together generate similar sampling distributions. Below we give specific choices for rigid and articulated robots. As we will soon see in Section 4.6, these heuristics worked well in our experiments.

For a rigid robot, the feature set is the union of two point sets, CH and MP. CH consists of the vertices on the convex hull of the robot. MP contains a single point in the middle of the robot, *e.g.*, the centroid. For an articulated robot, we take CH and MP with respect to each rigid link of the robot and take their union.

4.6 Implementation and Experiments

We implemented WCO using C++ and using the Qhull [Barber et al., 1996] library for workspace triangulation. Implementation details of the primitives, *i.e.*, `FreeConf`, `FreePath`, and `AllSolved` are presented in Appendix A.

For comparison purposes, we implemented the basic-PRM [Kavraki et al., 1996b] and the original Adaptive Hybrid Sampling (AHS) [Hsu et al., 2005] which combines uniform distribution, several Gaussian strategy [Boor et al., 1999] with different parameters, and several randomized bridge builder (RBB) [Sun et al., 2005] with different parameters. The basic-PRM is used as a benchmark of the difficulty of the problem. While the original-AHS is used because it is one of the most recent probabilistic planners that have been shown to perform well and its implementation is relatively simple. Furthermore, we also compared WCO with our simple workspace-based planner, WIS (Chapter 3).

4.6.1 Comparison with Other Planners

We tested WCO on several scenarios involving 2D and 3D workspace as well as rigid and articulated robot. These scenarios are the same as those used to test WIS. For ease of reading we show the scenarios again in Figure 4.3. To set the parameters of each planner, we select several representative values for the parameter(s). Each planner with different parameter values were ran 10 times independently on each test scenario. The parameter values that generate the best performance are then used as the parameter for testing the planner on the particular scenario. For testing, each planner were ran 30 times independently on each testing scenario. All experiments were conducted on a PC with a 3 GHz Intel Pentium 4 and 1 GB memory. The average results of the 30 runs are shown in Table 4.1.

In all tests, WCO used the convex hull vertices CH and the middle point MP of the robot as the feature point. For **Test 1–3**, CH refers to the convex hull vertices and MP refers to the centroid of the rigid body robot. However, **Test 4** uses a common articulated robot with a fixed base and all rotational joints such that the workspace displacement of the robot's links near the base is very limited.

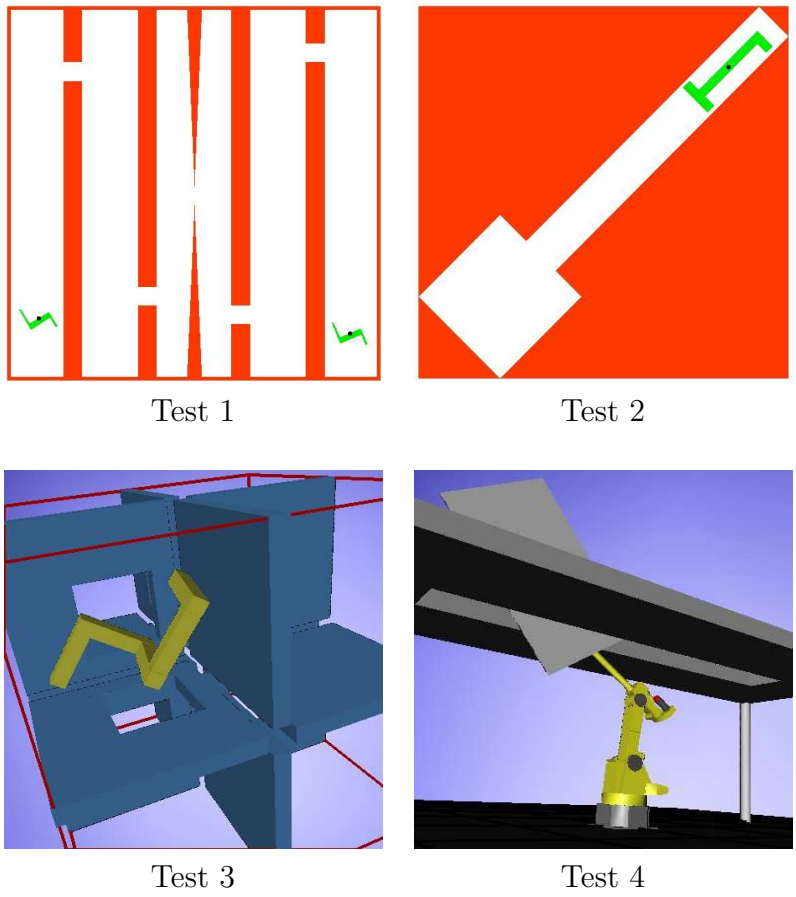


Figure 4.3: Test scenarios. **Test 1**: A 3-dofs rigid-body robot moves from the lower left corner to the lower right corner by passing through five narrow openings. **Test 2**: A 3-dofs rigid-body robot turns 180 degrees in a narrow dead-end. **Test 3**: A 6-dofs rigid-body robot must pass through 6 out of 7 narrow openings in order to answer the given query. **Test 4**: A 6-dofs robot manipulator with its end-effector holding a large plate maneuvers through a narrow slot.

To improve computational efficiency, we consider only the feature points in the furthest link, which contains the end-effector and the large plate. So, MP refers to the wrist point of the robot, while CH refers to the convex hull of the plate. In all tests, WIS used a single feature point, denoted as MP. For rigid body robot (**Test 1–3**), MP refers to the centroid and for an articulated robot (**Test 4**), MP refers to the wrist point.

Overall, WCO performed significantly faster than the original AHS and WIS (see Table 4.1). Although WCO incurs the additional costs of processing the workspace geometry and updating the sampling distribution, it uses fewer milestones and places them in strategic locations. It improves the overall performance

Table 4.1: Performance comparison of several probabilistic path planners. All times are measured in seconds.

Planner	**Test 1**				
	T_{pre}	T_{upd}	$T_{\text{tot}} \pm$ std	N_{mil}	N_{sam}
Basic-PRM			75.9 ± 45.1	13,540	52,687
Original AHS			23.0 ± 8.5	3,477	164,776
WIS	0.034		6.7 ± 2.0	1,660	7,024
WCO	0.045	0.072	2.8 ± 1.2	650	2,448
Planner	**Test 2**				
	T_{pre}	T_{upd}	$T_{\text{tot}} \pm$ std	N_{mil}	N_{sam}
Basic-PRM			4.1 ± 1.3	601	53,616
Original AHS			3.3 ± 1.3	163	76,742
WIS	0.007		0.7 ± 0.3	154	11,521
WCO	0.008	0.012	0.8 ± 0.6	170	5,531
Planner	**Test 3**				
	T_{pre}	T_{upd}	$T_{\text{tot}} \pm$ std	N_{mil}	N_{sam}
Basic-PRM			94.6 ± 48.6	9,011	36,594
Original AHS			56.7 ± 20.2	1,669	198,313
WIS	0.607		80.3 ± 30.0	5,723	160,686
WCO	0.942	2.408	25.9 ± 9.4	2,080	22,811
Planner	**Test 4**				
	T_{pre}	T_{upd}	$T_{\text{tot}} \pm$ std	N_{mil}	N_{sam}
Basic-PRM			69.8 ± 26.0	9,246	35,878
Original AHS			56.0 ± 27.9	2,672	168,013
WIS	0.071		200.7 ± 90.4	14,423	961,613
WCO	0.244	0.993	31.1 ± 13.1	3,211	62,405

T_{pre}: time for triangulating$\mathcal{W}_{\mathcal{F}}$.
T_{upd}: time for updating component sampling distributions (Algorithm 4.1, lines 4–12).
$T_{\text{tot}} \pm$ std: total running time ± standard deviation.
N_{mil}: number of milestones required for answering the query.
N_{sam}: number of configurations sampled.

by reducing the total number of collision checks needed for sampling new milestones and connecting milestones in the roadmap. See Figure 4.4 for an illustration of the differences between WCO and the other planners.

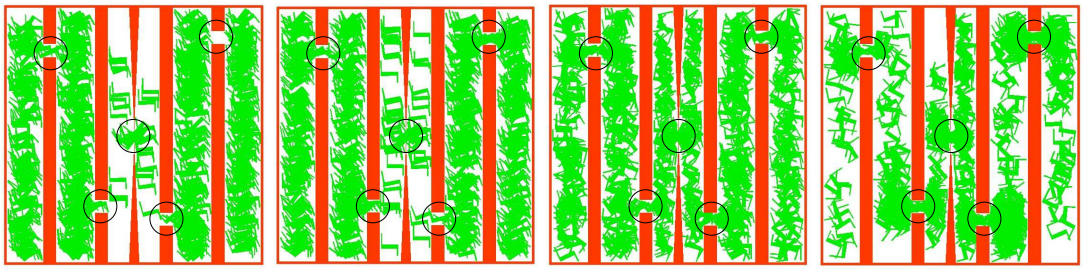
For comparison between WCO and the original AHS, it is especially interesting to consider **Test 2**. The start configuration q_i and the goal configuration q_g, when projected to $\mathcal{W}$, are very close. However, to go from q_i to q_g, the robot must go out of the narrow tunnel, reorient, and then go back to the tunnel again. Regardless of

which feature point a is chosen, it may potentially mislead the planner, because all short paths in $\mathcal{W}$ between $P_a(q_i)$ and $P_a(q_g)$ give little information on the correct configuration-space path that connects q_i and q_g. Nevertheless, WCO performed well here, because it combines information from both $\mathcal{W}$ and $\mathcal{C}$. It dynamically updates the workspace channels, which provide information for connecting distinct roadmap components. By doing so, as soon as $\mathcal{F}$ is covered adequately by $\mathcal{R}$, WCO can potentially identify the correct regions of $\mathcal{C}$ to sample.

Compared with WIS, WCO performed significantly better except for **Test 2**. This is expected, because WIS uses a single feature point (MP) and a static sampling distribution, which does not respond to changes in $\mathcal{R}$ and wastes lots of effort in sampling regions of $\mathcal{C}$ that are already well covered. Recall from Chapter 3 that the poor performance of WIS in **Test 3** is due to the many short triangles that actually correspond to forbidden regions. By combining workspace information with information from the configuration space, WCO is able to alleviate this false positive problem. Again recall from Chapter 3 that the poor performance of WIS in **Test 4** is caused by the insufficient representation of the robot and the many false positive cases due to the many short triangles that correspond to wide-open space in the robot's free-space. By using multiple feature points, instead of a single feature point, WCO is able to alleviate the problem with insufficient robot representation. We will discuss more about this as we assess the benefits of using multiple feature points in Section 4.6.2. Furthermore, by combining workspace information with information from the configuration space, WCO is able to alleviate the false positive problem. In **Test 2**, to solve the query, the entire $\mathcal{F}$ must be adequately covered, whether a static or a dynamic sampling distribution is used. WIS has an advantage, because it is simpler and does not incur the cost of updating the sampling distribution. Even so, the performance of WCO is comparable.

4.6.2 The Choice of Feature Points

To assess the benefits of multiple feature points, we ran WCO on **Test 1**–**4** with different feature sets. The experimental results show that although the performance of CH and MP varies across the test environments, the combined feature



Uniform Original AHS WIS WCO

Figure 4.4: The scenario is the same as **Test 1**. 700 milestones generated by different planners. Note that the width of the two chambers in the middle of the scene is smaller than the other four chambers. The pictures show that WCO increases the number of milestones in important regions that improve the connectivity of the roadmap, without generating too many unnecessary milestones in unimportant regions.

Table 4.2: The effect of feature points on the running times of WCO. All times are measured in seconds. |CH| denotes the number of feature points in CH.

Test Env.	|CH|	MP	CH	CH ∪MP
Test 1	6	2.2	4.7	2.7
Test 2	5	1.3	0.7	0.8
Test 3	13	40.8	28.9	25.9
Test 4	8	154.3	62.0	31.1

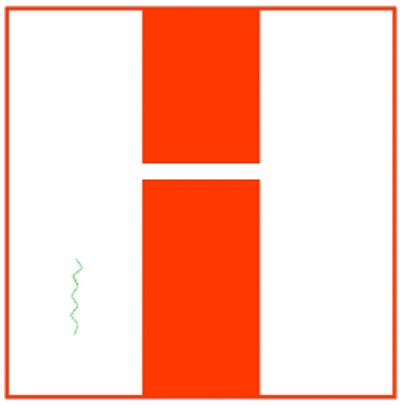
set CH∪MP has consistently good performance (see Table 4.2). This shows the usefulness of using multiple feature points.

However, notice that in **Test 1**, WCO with CH as the feature points performed worse than WCO with MP as the feature point. Although the number of feature points used in WCO with CH is more than that used in WCO with MP, its performance is worse. WCO with CH requires around 1,000 milestones with $\#milestones/\#samples \approx 0.28$, while that with MP requires only around 650 milestones with $\#milestones/\#samples \approx 0.23$. These numbers indicate that the centroid is a good representation of the robot. In this scenario, the workspace channels generated by different feature points are similar. But when the feature point a is a vertex of the convex hull, the subset $L_a(\mathcal{T}') \cap \mathcal{F}$ is relatively large and consists of wide-open regions that lie near the narrow passages of $\mathcal{F}$ and narrow regions that lie inside the narrow passages of $\mathcal{F}$. In this scenario, the narrow

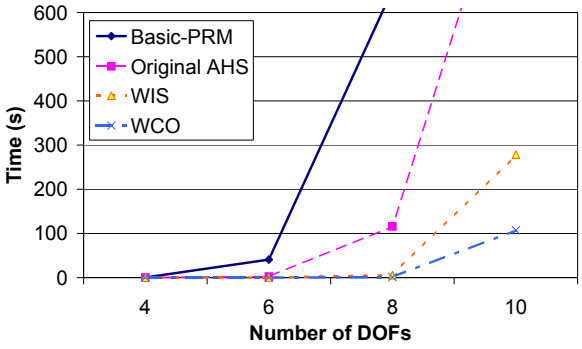
passages refer to configurations that will place most part of the robot in one of the five narrow openings. Surely, the configurations inside the narrow passages is more useful in solving the given query. However, since WCO does not bias sampling towards narrow regions in $L_a(\mathcal{T}')$, many of the sampled milestones fall in the wide open regions which are less useful for answering the given query. On the other hand, when the feature point a is MP of the robot, $L_a(\mathcal{T}') \cap \mathcal{F}$ is smaller, but most if not all of configurations in this subset lie inside the narrow passages of $\mathcal{F}$. So, although it is more difficult to sample a milestone, most of the sampled milestones are more useful. Therefore, WCO with CH requires more milestones and has higher $\#milestone/\#samples$ ratio. This means, WCO with MP as the feature point is able to place milestones at better locations than WCO with CH as the feature points. These results indicate that better choice of feature points may improve WCO's performance significantly.

Nevertheless, the experimental results in **Test 1–3** shows that when we combine both CH and MP as the feature points, WCO's performance is close to the best performance of either WCO with CH or WCO with MP as the feature points. This corroborates the theoretical result that the ensemble sampler is almost as good as the best component sampler and demonstrates the effectiveness of the AHS approach.

Test 4 shows that the performance of WCO with CH∪MP as the feature points is significantly better than either WCO with CH or WCO with MP as the feature points. **Test 4** requires a more coordinated robot motion to solve the given query. When more feature points are used, WCO considers more kinematic constraints of the robot, which means more robot's information is incorporated in generating the sampling distribution. By doing so, WCO is able to generate a more suitable sampling distribution. This strengthen our argument in Section 4.5.2 that multiple feature points implicitly takes the kinematic constraints of the robot into consideration.



Test 5



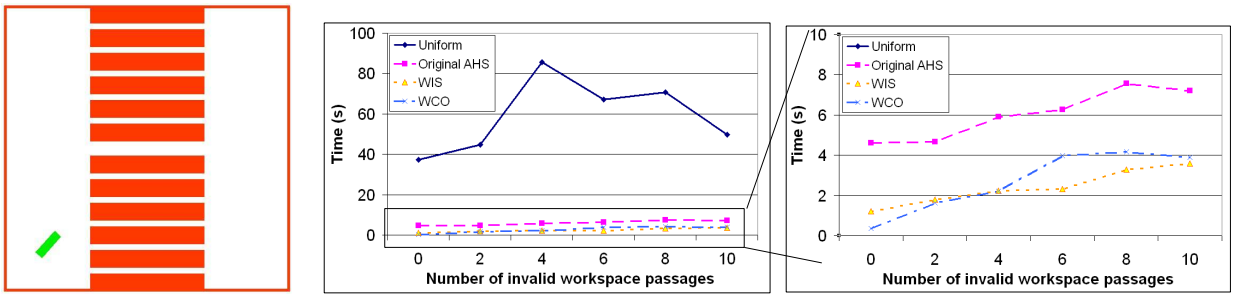
Results

Figure 4.5: Additional test for testing the performance of the planners as dim($\mathcal{C}$) increases. The robot is a planar articulated arm with a free-flying base. The dimensionality of $\mathcal{C}$ is increased by adding up to 8 links to the robot, resulting in a maximum of 10 dofs. The robot must move through the narrow passage in the middle.

4.6.3 Other Experiments

One concern of using workspace information to guide sampling in $\mathcal{C}$ is that as the dimensionality of $\mathcal{C}$ increases, workspace information becomes less useful. For this, we constructed a test environment with increasing dimensionality of $\mathcal{C}$ (**Test 5**). The results indicate that workspace information still has its merit (Figure 4.5). This corroborates our results in Section 3.3 that the usefulness of workspace information does not directly depend on the dimensionality of $\mathcal{C}$. Instead, it depends more on how close the visibility property of $\mathcal{W}_{\mathcal{F}}$ resembles that of $\mathcal{F}$.

One drawback of WCO is that it may find false workspace passages as channels, *i.e.*, workspace passages that the robot can not pass through. It seems plausible that as the number of false passages grows, the performance of WCO will keep on worsening. So, we performed a test with an increasing number of false workspace passages (**Test 6**). The results indicate that this trend happens, but only to a limited extent (Figure 4.6). Although the number of triangles and hence potential terminals in G' increases as the number of false workspace passages increases, after a certain limit, the number of workspace channels will not increase according



Test 6 Results

Figure 4.6: The performance of WCO, as the number of false passages increases. A 3-dofs rigid-body robot moves from the left to the right wide-open space. It only fits through the passage in the middle. The number of false passages increases from 2 to 10.

to the number of workspace triangles. Once the left and right wide-open space have been adequately covered, by construction (Algorithm 4.2, line 7–8), new sampled configurations that place the robot in the wide-open space will no longer generate new workspace channels. New workspace channels may be generated only when a configuration that lies in the narrow passage is sampled. Hence, when the solution is found after the wide-open space is covered adequately, increasing false workspace passages will not cause additional workspace channels. As a result, after a certain limit, increasing the number of false workspace passage will no longer worsen WCO's performance.

We have also experimented with a more complicated problem. In the attached video demo, we show a bridge inspection scenario involving a 35-dofs robot. The robot consists of 11 links where each link is attached to a spherical joint. The base of the robot can translate in 2 dimensions. A camera is attached to the end-link of the robot. The goal is to take a picture of four locations of the bridge. The robot's configurations for taking the first two and the last two pictures are similar, the difference is only in the position of the robot's base. To achieve the goal, we ran the planner to solve 3 queries. First, the robot needs to move from a default configuration, where all its parts are located below the bridge, to a stretched configuration such that it can take the first picture. To solve this query, the planner needs to sample narrow passages due to the small opening on the bridge and narrow passages due to the singularity of the robot. Second, after taking the

first picture, the robot needs to move to take the second picture. The narrow passage in this query is mainly due to the singularity of the robot. The last query requires the robot to move from one default position to another. WCO uses the centroid of the camera, the centroid of each spherical joint, and the centroid of the base as its feature points. WIS uses only the centroid of the camera as its feature points. We ran WCO for 5 times and in all the runs, WCO solves the problem in less than an hour. But, basic-PRM, original AHS, and WIS fail to solve the bridge inspection problem after a 24 hours run. These methods fail to find a solution for the first two queries.

The good performance of WCO is mainly due to the use of workspace channels and multiple feature points, which enable WCO to focus its search in the more useful subset of $\mathcal{C}$. By restricting the search only to the lift-mapping of the workspace channels, WCO samplers significantly reduce the search to a subset of $\mathcal{C}$ that is more likely to connect the initial and goal configurations of the given queries. Nevertheless, given the high dimensionality of $\mathcal{C}$, the reduced search space is still large. Using multiple feature points and adaptive weighting, WCO assigns higher sampling distribution to subsets of the lift-mapping of $\mathcal{C}$ that satisfy the robot's kinematics constraints (see Section 4.5.2). Hence, enabling WCO to further focus its search in subsets of $\mathcal{C}$ that is more likely to be valid and useful for solving the given queries.

4.7 Analysis

Now, we will theoretically analyze the performance of WCO. We start by deriving the sampling density of WCO in Section 4.7.1. We then present the probabilistic completeness and running time of WCO in Section 4.7.2 and Section 4.7.3 respectively.

4.7.1 Sampling Density

To derive the sampling density generated by WCO, we will start by deriving the sampling density generated by each component sampler at a particular time. In

this context, time is discrete. One unit time refers to the steps for sampling a configuration. We then compute the sampling density of WCO at a particular time.

To derive the sampling density of the component sampler $s_i, i \in [0, L-1]$ at a particular time j, we need to look at the two-stage sampling procedure (Section 4.4.3) and the probability density function used in each of the sampling stage. In the first stage, s_i samples a point $\mathbf{x} \in \mathcal{W}_\mathcal{F}$ by sampling a triangle $t \in \mathcal{T}_\mathcal{F}$ according to the current probability mass function which is defined as

$$p_{T_j|S_j}(t \mid s_i) = \begin{cases} \mu(t) & i = 0 \\ \frac{1}{|\mathcal{T}'_{j,s_i}|} & i \in [1, L-1] \ , \ t \in \mathcal{T}'_{j,s_i} \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where T_j and S_j are the random variables that represent the triangle and sampler picked at time j, $\mu(.)$ denotes volume, $\mathcal{T}'_{j,s_i}$ denotes the workspace channels used by s_i to sample a configuration at time j, and $|\mathcal{T}'_{j,s_i}|$ denotes the number of triangles in $\mathcal{T}'_{j,s_i}$. The sampler s_i is the uniform component sampler when $i = 0$, and the WCO component sampler when $i \in [1, L-1]$. Furthermore, we assume that $\sum_{t \in \mathcal{T}_\mathcal{F}} \mu(t) = 1$. After a triangle t is sampled, s_i samples a point $\mathbf{x} \in t$ uniformly at random. So, the conditional density function for sampling a point from a triangle t is:

$$f_{X_j|T_j,S_j}(\mathbf{x} \mid t, s_i) = \begin{cases} \frac{1}{\mu(t)} & \mathbf{x} \in t \ , \ t \in \mathcal{T}_\mathcal{F} \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

where X_j is the random variable that represents a point sampled from $\mathcal{W}_\mathcal{F}$ at time j. The density function used by s_i to sample a point $\mathbf{x}$ in $\mathcal{W}_\mathcal{F}$ at time j can then be computed as

$$f_{X_j|S_j}(\mathbf{x} \mid s_i) \ = \ \sum_{t \in \mathcal{T}_\mathcal{F}} f_{X_j|T_j}(\mathbf{x} \mid t, s_i) p_{T_j|S_j}(t \mid s_i)$$

$$= \begin{cases} 1 & i = 0 \\ \frac{1}{|\mathcal{T}'_{j,s_i}| \cdot \mu(\tau(\mathbf{x}))} & i \in [1, L-1] \ , \ \tau(\mathbf{x}) \in \mathcal{T}'_{j,s_i} \\ 0 & \text{otherwise} \end{cases} \quad (4.6)$$

where $\tau(\mathbf{x})$ is the triangle that contains $\mathbf{x}$.

In the second stage, after $\mathbf{x}$ has been sampled, s_i samples a configuration uniformly at random from $L_a(\mathbf{x})$. So, at time j, the density function used by s_i to sample $\mathcal{C}$ given a point $\mathbf{x} \in \mathcal{W}_{\mathcal{F}}$, is defined as

$$f_{Q_j|X_j,S_j}(q \mid \mathbf{x}, s_i) = \begin{cases} \frac{1}{\lambda(L_a(\mathbf{x}))} & q \in L_a(\mathbf{x}) \ , \ \mathbf{x} \in t \ , \\ & \quad \text{if } i = 0 \ , \ t \in \mathcal{T}_{\mathcal{F}} \ ; \text{ if } i \in [1, L-1] \ , \ t \in \mathcal{T}'_{j,s_i} \\ 0 & \text{otherwise} \end{cases} \quad (4.7)$$

where $\lambda(\cdot)$ represents the volume in the $(n - \dim(\mathcal{W}))$-dimensional subspace of $\mathcal{C}$. Assuming that $\mathcal{C}$ is a normalized Euclidean space $[0,1]^{\dim(\mathcal{C})}$, the largest possible value of $\lambda(L_a(\mathbf{x}))$ is one.

The sampling density generated by a WCO component sampler s_i at time j can then be computed as

$$f_{Q_j|S_j}(q \mid s_i) = \int_{\mathbf{x} \in \mathcal{W}_{\mathcal{F}}} f_{Q_j|X_j,S_j}(q \mid \mathbf{x}, s_i) f_{X_j|S_j}(\mathbf{x} \mid s_i) \partial \mathbf{x} \quad (4.8)$$

$$= \begin{cases} \frac{1}{\lambda(L_a(Pa(q)))} & i = 0 \\ \frac{1}{\lambda(L_a(Pa(q)))} \cdot \frac{1}{|\mathcal{T}'_{j,s_i}| \cdot \mu(\tau(Pa(q)))} & i \in [1, L-1] \ , \\ & \quad \tau(Pa(q)) \in \mathcal{T}'_{j,s_i} \\ 0 & \text{otherwise} \end{cases} \quad (4.9)$$

And the sampling density generated by WCO at time j can then be computed as

$$f_{Q_j}(q) = \sum_{i=0}^{L-1} f_{Q_j|S_j}(q \mid s_i) p_{S_j}(s_i) \quad (4.10)$$

where $f_{Q_j|S_j}(q \mid s_i)$ is defined in (4.9) and $p_{S_j}(s_i)$ is defined in (4.1).

4.7.2 Probabilistic Completeness

Intuitively, it is obvious that WCO is probabilistically complete because one of WCO's component samplers, *i.e.*, the uniform component sampler s_0, is probabilistically complete and $p_{S_j}(s_0)$ is always positive for any time j. This means, given a long enough time, WCO will eventually find a path between the given queries if one exists.

However, having derived the sampling density of WCO, we can use any analysis of the basic-PRM, *e.g.*, [Kavraki et al., 1995,Kavraki et al., 1996a,Hsu et al., 1997] to better analyze WCO's performance. Below, we show an example of adapting the analysis in [Kavraki et al., 1996a] to analyze WCO. The result shows that even when no channels predict the path correctly, the probability that WCO does not find a path between the given queries converges to zero exponentially in term of the number of milestones, provided such a path exists. Furthermore, if the channels predict the path between the given query well and each triangle inside the channels has volume less than the average volume of the triangles in $\mathcal{T}_{\mathcal{F}}$, then the upper bound of the failure probability of WCO is lower than that of the basic-PRM.

Suppose WCO consists of L component samplers: $\mathcal{S} = \{s_0, \ldots, s_{L-1}\}$, where s_0 is the uniform component sampler and the rest are the WCO component samplers. And suppose $p_{S_j}(s_i)$ is the probability of using s_i to sample a configuration at time j. Then, the performance of WCO can be stated more formally as

Theorem 4.3 *Let ψ be a collision-free path that solves the given query. Suppose l is the length of ψ, d is the shortest distance between a configuration in ψ to the forbidden region, and $\mathcal{B}_{\frac{d}{2}}$ is any ball of radius $\frac{d}{2}$ in $\mathcal{F}$. Let's denote $\mathcal{Q}_\psi$ as the set of all balls of radius $\frac{d}{2}$ centered at each configuration in ψ. For a particular time j, let's denote $\mathcal{K}_j = \min_{\mathcal{B}\in\mathcal{Q}_\psi} \mathcal{K}_j(\mathcal{B})$, where $\mathcal{K}_j(\mathcal{B})$ is the smallest (among all WCO component samplers) lower bound of the probability that a WCO component sampler samples a configuration from ball $\mathcal{B}$. Then, the probability that ψ can not be found using a roadmap of N milestones, constructed by a WCO planner, is* $Pr(failure) \leq \left(\lceil \frac{2l}{d} \rceil - 1\right) \prod_{j=1}^{N} \left(1 - \left(\mu(\mathcal{B}_{\frac{d}{2}}) p_{S_j}(s_0) + \mathcal{K}_j \cdot \sum_{i=1}^{L-1} p_{S_j}(s_i)\right)\right)$.

Proof Suppose $B = \lceil \frac{2l}{d} \rceil$. We can discretize ψ into a sequence of configurations $q^0, \ldots, q^B$ such that for $k \in [0, B]$, the length of ψ between q^k and q^{k+1} is at

most $\frac{d}{2}$, $q^0 = q_i$, and $q^B = q_g$. Then, ψ can be covered with balls $\mathcal{B}(q^k, \frac{d}{2})$ of radius $\frac{d}{2}$ and center at q^k. An illustration is shown in Figure 3.9. Notice that $\mathcal{B}(q^{k+1}, \frac{d}{2}) \subseteq \mathcal{B}(q^k, d)$. Hence, if the roadmap constructed by WCO places at least one milestone in each ball, then the generated roadmap finds the path. So, the probability that WCO fails to find a path after the generated roadmap contains N milestones can be written as,

$$\begin{aligned} Pr(failure) &\leq Pr[\exists k\ \mathcal{B}(q^k, \tfrac{d}{2}) \text{ is empty}] \\ &\leq \sum_{k=1}^{B-1} Pr[\mathcal{B}(q^k, \tfrac{d}{2}) \text{ is empty}] \end{aligned} \tag{4.11}$$

Given the workspace channels and the probability distribution for choosing a component sampler at each time j, the probability of sampling a configuration at a particular time is independent of the probability of sampling a configuration at the preceding time, and we can expand (4.11) to:

$$\begin{aligned} Pr(failure) &\leq \sum_{k=1}^{B-1} \prod_{j=1}^{N} (1 - Pr[Q_j \in \mathcal{B}(q^k, \tfrac{d}{2})]) \\ &= \sum_{k=1}^{B-1} \prod_{j=1}^{N} \left(1 - \int_{q \in \mathcal{B}(q^k, \frac{d}{2})} f_{Q_j}(q) \partial q\right) \\ &= \sum_{k=1}^{B-1} \prod_{j=1}^{N} \left(1 - \int_{q \in \mathcal{B}(q^k, \frac{d}{2})} \sum_{i=0}^{L-1} f_{Q_j|S_j}(q \mid s_i) p_{S_j}(s_i) \partial q\right) \end{aligned}$$

Reordering the terms gives us:

$$= \sum_{k=1}^{B-1} \prod_{j=1}^{N} \left(1 - \sum_{i=0}^{L-1} p_{S_j}(s_i) \int_{q \in \mathcal{B}(q^k, \frac{d}{2})} f_{Q_j|S_j}(q \mid s_i) \partial q\right) \tag{4.12}$$

Let's first compute the probability that a component sampler $s_i, i \in [0, L-1]$ samples a configuration in $\mathcal{B}(q^k, \frac{d}{2})$ at time j. We compute this value separately for WCO component samplers, *i.e.*, $s_i, i \in [1, L-1]$, and for the uniform component sampler, *i.e.*, s_0, based on the density function in (4.9). For a WCO component

sampler, *i.e.*, $s_i, i \in [1, L-1]$, we have

$$\int_{q \in \mathcal{B}(q^k, \frac{d}{2})} f_{Q_j|S_j}(q \mid s_i)\partial q \;=\; \int_{q \in \mathcal{B}(q^k, \frac{d}{2})} \frac{1}{\lambda(L_a(P_a(q)))} \cdot \frac{1}{|\mathcal{T}'_{j,s_i}| \cdot \mu(\tau(P_a(q)))}$$

Since q must lie in $L_a(\mathbf{x})$ and $\mathbf{x}$ must lie in t where t is in $\mathcal{T}'_{j,s_i}$, the right hand side of the above equation can be written as

$$\begin{aligned} &= \sum_{t \in \mathcal{T}'_{j,s_i}} \int_{\mathbf{x} \in t} \int_{q \in \Big(\mathcal{B}(q^k, \frac{d}{2}) \cap L_a(\mathbf{x})\Big)} \frac{1}{\lambda(L_a(\mathbf{x})) \cdot |\mathcal{T}'_{j,s_i}| \cdot \mu(t)} \partial q \partial \mathbf{x} \\ &= \sum_{t \in \mathcal{T}'_{j,s_i}} \int_{\mathbf{x} \in t} \frac{\lambda(\mathcal{B}(q^k, \frac{d}{2}) \bigcap L_a(\mathbf{x}))}{\lambda(L_a(\mathbf{x})) \cdot |\mathcal{T}'_{j,s_i}| \cdot \mu(t)} \partial q \partial \mathbf{x} \end{aligned}$$

Taking the maximum possible value for $\lambda(L_a(\mathbf{x}))$, we get:

$$\begin{aligned} &\geq \sum_{t \in \mathcal{T}'_{j,s_i}} \int_{\mathbf{x} \in t} \frac{\lambda(\mathcal{B}(q^k, \frac{d}{2}) \bigcap L_a(\mathbf{x}))}{|\mathcal{T}'_{j,s_i}| \cdot \mu(t)} \partial \mathbf{x} \\ &= \sum_{t \in \mathcal{T}'_{j,s_i}} \frac{\mu(\mathcal{B}(q^k, \frac{d}{2}) \bigcap L_a(t))}{|\mathcal{T}'_{j,s_i}| \cdot \mu(t)} \\ &\geq \left(\frac{1}{|\mathcal{T}'_{j,s_i}|} \min_{t \in \mathcal{T}'_{j,s_i}} \frac{1}{\mu(t)} \right) \mu(\mathcal{B}(q^k, \tfrac{d}{2}) \bigcap L_a(\mathcal{T}'_{j,s_i})) \\ &= \mathcal{K}^i_j \cdot \mu(\mathcal{B}(q^k, \tfrac{d}{2}) \bigcap L_a(\mathcal{T}'_{j,s_i})) \end{aligned} \tag{4.13}$$

where $\mathcal{K}^i_j = \frac{1}{|\mathcal{T}'_{j,s_i}|} \min_{t \in \mathcal{T}'_{j,s_i}} \frac{1}{\mu(t)}$.

Now for the uniform component sampler s_0, the probability that s_0 samples a configuration in $\mathcal{B}(q^k, \frac{d}{2})$ at time j can be written as

$$\int_{q \in \mathcal{B}(q^k, \frac{d}{2})} f_{Q_j|S_j}(q \mid s_i)\partial q \;=\; \int_{q \in \mathcal{B}(q^k, \frac{d}{2})} \frac{1}{\lambda(L_a(P_a(q)))}$$

Since q must lie in $L_a(\mathbf{x})$ where $\mathbf{x} \in \mathcal{W}_{\mathcal{F}}$, the right hand side of the above equation can be written as

$$= \int_{\mathbf{x} \in \mathcal{W}_{\mathcal{F}}} \int_{q \in \Big(\mathcal{B}(q^k, \frac{d}{2}) \cap L_a(\mathbf{x})\Big)} \frac{1}{\lambda(L_a(\mathbf{x}))} \partial q \partial \mathbf{x}$$

And taking the maximum possible value for $\lambda(L_a(\mathbf{x}))$ gives us:

$$\begin{aligned} &\geq \int_{\mathbf{x}\in\mathcal{W}_{\mathcal{F}}} \lambda((\mathcal{B}(q^k, \tfrac{d}{2}) \bigcap L_a(\mathbf{x})))\partial q \\ &= \mu(\mathcal{B}(q^k, \tfrac{d}{2})) \end{aligned} \tag{4.14}$$

So, the probability that a component sampler $s_i, i \in [0, L-1]$ at time j samples a configuration in $\mathcal{B}(q^k, \frac{d}{2})$ is:

$$\int_{q\in\mathcal{B}(q^k,\frac{d}{2})} f_{Q_j|S_j}(q \mid s_i)\partial q \;\geq\; \begin{cases} \mu(\mathcal{B}(q^k, \frac{d}{2})) & i = 0 \\ \mathcal{K}_j^i \cdot \mu(\mathcal{B}(q^k, \frac{d}{2}) \bigcap L_a(\mathcal{T}'_{j,s_i})) & i \in [1, L-1] \end{cases} \tag{4.15}$$

Substituting (4.15) into (4.12) gives us

$$Pr(failure) \;\leq\; \sum_{k=1}^{B-1}\prod_{j=1}^{N}\left(1 - \left(p_{S_j}(s_0)\cdot\mu(\mathcal{B}(q^k,\tfrac{d}{2})) + \sum_{i=1}^{L-1} p_{S_j}(s_i)\cdot\mathcal{K}_j^i\cdot\mu(\mathcal{B}(q^k,\tfrac{d}{2})\bigcap L_a(\mathcal{T}'_{j,s_i}))\right)\right)$$

Setting $\mathcal{K}_j(\mathcal{B}(q^k, \frac{d}{2}))$ as the smallest lower bound of the probability that a WCO component sampler samples a configuration from $\mathcal{B}(q^k, \frac{d}{2})$ at time j, *i.e.*, $\mathcal{K}_j(\mathcal{B}(q^k, \frac{d}{2})) = \min_{i=1}^{L-1} \mathcal{K}_j^i \cdot \mu(\mathcal{B}(q^k, \frac{d}{2}) \bigcap L_a(\mathcal{T}'_{j,s_i}))$, gives us:

$$Pr(failure) \;\leq\; \sum_{k=1}^{B-1}\prod_{j=1}^{N}\left(1 - \left(p_{S_j}(s_0)\cdot\mu(\mathcal{B}(q^k,\tfrac{d}{2})) + \mathcal{K}_j(\mathcal{B}(q^k,\tfrac{d}{2}))\cdot\sum_{i=1}^{L-1} p_{S_j}(s_i)\right)\right)$$

And since $\min_{k=1}^{B-1} \mathcal{K}_j(\mathcal{B}(q^k, \frac{d}{2})) \geq \mathcal{K}_j$, we have

$$Pr(failure) \;\leq\; \left(\left\lceil\frac{2l}{d}\right\rceil - 1\right)\prod_{j=1}^{N}\left(1 - \left(\mu(\mathcal{B}_{\frac{d}{2}})p_{S_j}(s_0) + \mathcal{K}_j\cdot\sum_{i=1}^{L-1} p_{S_j}(s_i)\right)\right).$$

which is the result we want. □.

Notice that even when all workspace channels of WCO are misleading and hence $\mathcal{K}_j = 0$ for all time j, the failure probability of WCO still converges to zero exponentially in the number of milestones, provided a solution exists. The reason is that although $\mathcal{K}_j \geq 0$, the value $\left(\mu(\mathcal{B}_{\frac{d}{2}})p_{S_j}(s_0) + \mathcal{K}_j \cdot \sum_{i=1}^{L-1} p_{S_j}(s_i)\right)$ is strictly greater than zero, because (4.1) guarantees that $p_{S_j}(s_0) > \frac{\gamma}{K}$ where $\gamma \in (0, 1]$.

Of course when all workspace channels of WCO are misleading, compared to the basic-PRM where the upper bound of its failure probability is $\lceil \frac{2l}{d} \rceil (1 - \mu(\mathcal{B}_{\frac{d}{2}}))^N$, the upper bound of the failure probability of WCO is higher. This is reasonable as when all workspace channels are misleading, WCO spends more time for sampling parts of the configuration space that does not help in solving the given query.

However, when $\mathcal{K}_j$ is larger than $\mu(\mathcal{B}_{\frac{d}{2}})$ for all time j, the upper bound of the failure probability of WCO is guaranteed to be lower than that of the basic-PRM, *i.e.*, $\left(\lceil \frac{2l}{d} \rceil - 1\right) \left(1 - \mu(\mathcal{B}_{\frac{d}{2}})\right)^N$. Two components that determine how large $\mathcal{K}_j$ is, are $\mathcal{K}_j^i$ and $\mu(\mathcal{B}(q^k, \frac{d}{2}) \bigcap La(\mathcal{T}'_{j,s_i}))$ of the WCO component samplers $s_i, i \in [1, L]$ and configurations q^k in ψ.

When the channels predict the path perfectly at all time, *i.e.*, the workspace channels of any WCO component samplers at any time j enclose $Pa(\bigcup_{q \in Q_\psi} \mathcal{B}(q, \frac{d}{2}))$, then $\mu(\mathcal{B}(q^k, \frac{d}{2}) \bigcap La(\mathcal{T}'_{j,s_i})) = \mu(\mathcal{B}_{\frac{d}{2}})$ for any time j, any sampler s_i, and any configuration q^k in ψ. This means, for $\mathcal{K}_j$ to be larger than $\mu(\mathcal{B}_{\frac{d}{2}})$, we only need $\mathcal{K}_j^i$ to be larger than 1 for any sampler $s_i, i \in [1, L-1]$. Assuming that $\sum_{t \in \mathcal{T}_\mathcal{F}} \mu(t) = 1$, the average volume $\overline{\mu_{\mathcal{T}'}}$ of the triangles in $\mathcal{T}_\mathcal{F}$ is $\frac{1}{|\mathcal{T}_\mathcal{F}|}$. This means that when the volume of each triangle in $\mathcal{T}'_{j,s_i}$ is less than the average volume $\overline{\mu_{\mathcal{T}'}}$, $\mathcal{K}_j^i > \frac{|\mathcal{T}_\mathcal{F}|}{|\mathcal{T}'_{j,s_i}|}$. And since any workspace channels is a subset of the triangles $\mathcal{T}_\mathcal{F}$ in $\mathcal{W}_\mathcal{F}$, $|\mathcal{T}_\mathcal{F}| \geq |\mathcal{T}'_{j,s_i}|$. Thus, when the volume of each triangle in $\mathcal{T}'_{j,s_i}$ is less than the average volume, $\mathcal{K}_j^i > 1$. Therefore when the channels generated by each WCO component samplers predict the path between the given query perfectly at all time and contain only triangles with volume smaller than the average triangle volume in $\mathcal{T}_\mathcal{F}$, the upper bound of the failure probability of WCO is guaranteed to be lower than that of the basic-PRM. Notice that triangles with volume smaller than the average indicates that the triangles are surrounded by nearby obstacles which suggests that the triangles correspond to narrow passages in $\mathcal{F}$. Hence, our analysis suggests that when ψ lie in the narrow passages of $\mathcal{F}$ and WCO predicts the path perfectly at all time, the upper bound of the failure probability of WCO tends to be lower than that of the basic-PRM.

Furthermore, when the channels are not perfect, but $\forall_{q \in \psi}\ 0 < \mu(\mathcal{B}(q, \frac{d}{2}) \bigcap La(\mathcal{T}'_{j,s_i})) < \mu(\mathcal{B}_{\frac{d}{2}})$, we can still have $\mathcal{K}_j$ that is larger than $\mu(\mathcal{B}_{\frac{d}{2}})$ if $\mathcal{K}_j^i$ is larger than $\dfrac{\mu(\mathcal{B}_{\frac{d}{2}})}{\mu(\mathcal{B}(q, \frac{d}{2}) \bigcap La(\mathcal{T}'_{j,s_i}))}$

for any WCO component sampler s_i at any time j. Assuming that the channels contain only triangles with volume smaller than the average triangle volume, larger value of $\mathcal{K}_j^i$ can be achieved if the workspace channels contain less triangles. Less number of triangles in the workspace channels means higher ratio of $\frac{|\mathcal{T}_{\mathcal{F}}|}{|\mathcal{T}'_{j,s_i}|}$, and hence $\mathcal{K}_j^i$ is more likely to be larger too. This result strengthen our preference of constructing short channels, as it is more likely to reduce the upper bound of the failure probability of WCO.

It is interesting to note that to get larger value of $\mathcal{K}_j^i$, in order to get larger value of $\mathcal{K}_j$, it is preferable to choose channels with small triangles. This seems odd as small triangles indicate that the triangles lie in narrow workspace passages. However, recall that in this analysis, we have determined the path ψ, and hence the set $\mathcal{Q}_\psi$ has been determined too. This means, regardless of the workspace channels, our goal is to sample the same set of $\mathcal{Q}_\psi$. So, suppose $\mathcal{T}'$ and $\mathcal{T}''$ are two possible workspace channels where $\forall_{\mathcal{B}\in\mathcal{Q}_\psi}\mu(\mathcal{B}\cap L_a(\mathcal{T}')) = \mu(\mathcal{B}\cap L_a(\mathcal{T}''))$. Then, it is more difficult to sample a configuration from $\mathcal{Q}_\psi$ when the workspace channels are larger. Therefore, it is reasonable to prefer channels that consists of small triangles and channels with a small number of triangles.

To see how the above result and comparisons hold in reality, we ran an experiment to test the convergence rate of WCO and compared it with the convergence rate of the basic-PRM. We use the robot and environement in Test 3. However, in this test we used 7 queries. Each query requires the robot to pass through one of the seven narrow passages in the environment. Different queries require the robot to pass through different narrow passages. For this purpose, we ran WCO and the basic-PRM planner with the same parameters as the one used in Section 4.6. Each planner was run to generate a roadmap with N number of milestones, where $N = 250; 500; \ldots; 20,000$. For each planner and each N, the planner was run for 30 times. Then, for each planner and each N, we compute the average failure rate, *i.e.*, the average number of queries that can not be answered by the roadmap with N milestones, over the 30 runs. The results are presented in Figure 4.7. This experiment shows that the failure rate of WCO converges to 0 around 3–4 times faster than the basic-PRM.

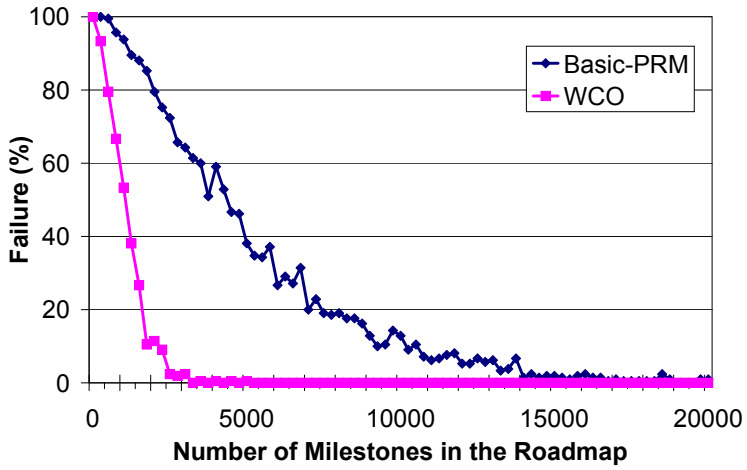


Figure 4.7: Failure rate as #milestones increases. The robot and environment are the same as the one used in **Test 3**. But in this experiment, we use 7 queries. Each query requires the robot to pass through one of the seven narrow passages in the environment. The failure rate is the percentage of the queries that can not be solved by the roadmap.

4.7.3 Running Time

As in WIS, the total running time of WCO consists of two main parts, *i.e.*, the time for extracting workspace information and the time for building the roadmap. WCO extracts workspace information only once, before roadmap construction. This information will then be copied to each component sampler. To extract workspace information, WCO runs five main steps. First, sampling $|P|$ points on the boundary of $\mathcal{W}_{\mathcal{F}}$, which takes $O(|P|)$ time. Second, triangulating the sampled points to generate $\mathcal{T}$. The cost of triangulation is $O(|P|^2)$ in the worst case, but in practice, we can expect $O(|P| \lg |P|)$ [Amenta et al., 2001]. The third step is separating the triangles inside $\mathcal{W}_{\mathcal{F}}$ from those outside and hence generate $\mathcal{T}_{\mathcal{F}}$. Suppose $|\mathcal{T}|$ is the number of triangles in $\mathcal{T}$. This computation takes $O(|\mathcal{T}|)$. In the fourth step, WCO computes and assigns the volume of each triangle in $\mathcal{T}_{\mathcal{F}}$. The volume will be used by the uniform component sampler of WCO. This computation takes linear time in the number of triangles in $\mathcal{T}_{\mathcal{F}}$. Lastly, WCO copies this workspace information to each of its component sampler. If WCO uses L component samplers, then this step takes $O(L)$. In general L is much lower

than $|\mathcal{T}|$. The number of triangles $|\mathcal{T}|$ is $O(|P|)$ for 2D workspace and $O(|P|^2)$ for 3D workspace. So, the total worst case time T_{pre} for pre-processing the workspace information is $O(|P|^2)$.

The second part of WCO running time is roadmap construction. This part iteratively performs three main steps. The first step is sampling a collision-free configuration. Let's denote the time used by WCO to generate a milestone as T_m. Compared to the basic-PRM, WCO takes slightly more time to sample a configuration because it needs to first select a component sampler, and then sample $\mathcal{C}$ using workspace information. For rigid body robot, the additional time is dominated by computing a configuration given a sampled workspace point (see Section 4.4.3). While for articulated robot, the additional time is dominated by the inverse kinematics (IK) computation. The second step is adding and connecting the new milestone to the milestones in the current roadmap. We denote this cost as T_l. The time taken for WCO to perform this step is the same as that taken by the basic-PRM. The last step is updating the sampling distribution. This step consists of two stages. First is adapting the sampling distribution of each WCO component sampler. For this, WCO component samplers start by updating the channel graph and workspace channels. This computation is quite efficient. Since we sample $\mathcal{C}$ based on workspace information, we can project the new milestone (Algorithm 4.2, line 4) in constant time. A loose upper bound for updating the label sets, G', and $\mathcal{T}'$ (Algorithm 4.2, lines 5–12) is $O(|\mathcal{T}_{\mathcal{F}}|)$. In practice, the upper bound is rarely reached. This stage is performed by each WCO component sampler. The second stage is updating the sampling distribution assigned to each sampler. This stage takes $O(L)$ where L is the number of component samplers of WCO and is in general much lower than $|\mathcal{T}_{\mathcal{F}}|$. The entire update step takes little time, compared to other parts of the planner, as we have seen in Section 4.6. Suppose T_u denotes the entire update time. The total time for WCO to build a roadmap of N_{mil} milestones is then $O(N_{\text{mil}} \cdot (T_m + T_l + T_u))$. And the total running time complexity of WCO to generate a roadmap with N_{mil} milestones is $O(T_{\text{pre}} + N_{\text{mil}}(T_m + T_l + T_u))$.

Let's now compare the running time of WCO with that of the basic-PRM.

The following method of comparing the running time of different sampling strategies have been used in [Sun et al., 2005]. Compared to the basic-PRM, WCO performs additional computation to extract workspace information and adapt its sampling distribution. As we have seen in Section 4.6, in general this cost is very small compared to the total cost. Furthermore, to sample a configuration using information from $\mathcal{W}_{\mathcal{F}}$, WCO requires an additional small constant time c. Suppose to generate a milestone, we need N_s samples. To simplify comparison, let's assume that the number of samples needed to generate a milestone is the same everywhere. The total time T_{wco} used by WCO to generate a roadmap of N_{mil} milestones is then $T_{wco} = T_{\text{pre}} + N_{\text{mil}}(N_s(T_s + c) + T_l + T_u)$, where T_s is the time used by the basic-PRM to generate a sample. While the total time T_{uni} used by the basic-PRM is $T_{uni} = N_{\text{mil}}(N_s \cdot T_s + T_l)$. However, WCO places its milestones at more important regions of $\mathcal{F}$ and hence uses less number of milestones to solve the given queries. Since in general T_l is much larger than T_s, then the total time for WCO to generate a roadmap that can solve the given queries is faster than the total time required by the basic-PRM. As an example, suppose to solve the given queries, the basic-PRM uses 5000 milestones while WCO uses 1000 milestones. And suppose $T_{\text{pre}} = 400T_s$, $N_s = 2$, $c = T_s$, $T_l = 10T_s$, and $T_u = 5T_s$. Then $T_{wco}/T_{uni} = (400 + 1000 \cdot (2 \cdot 2 \cdot T_s + 10 \cdot T_s + 5T_s))/(5000 \cdot (2 \cdot T_s + 10 \cdot T_s))$. Which results in $T_{wco} \approx 0.33T_{uni}$. So, compared to the basic-PRM, WCO pays a slightly higher cost to place milestones at more useful regions of $\mathcal{F}$, such that the total milestones needed for solving the given queries are less and hence the total time for solving the queries are less too.

4.8 Discussion

In this chapter, we have presented a new probabilistic path planner, called WCO, that uses workspace information along with sampling history and the current state of the roadmap to dynamically adapt the sampling distribution. WCO is composed of many component samplers, each based on a feature point of a robot. Each component sampler uses workspace information to estimate regions of $\mathcal{F}$ that is more

likely to improve the quality of the current roadmap. Using the adaptive hybrid sampling approach, WCO combines the component samplers. It uses sampling history to favor component samplers that have been performing well in the past. Our analysis shows that the failure probability of WCO converges to zero exponentially in the number of milestones, whenever a solution exists. And when WCO's estimation is good, the upper bound on the failure probability of WCO is lower than that of the basic-PRM. Our experimental results show that WCO performs up to 28 times faster than the basic-PRM and 8 times faster than recent probabilistic path planner that has been shown to perform well. And as the dimensionality of $\mathcal{C}$ increases, WCO's performance decreases slower than other recent probabilistic path planner. We have also successfully implemented WCO in a simulated bridge inspection scenario involving a 35-dofs robot.

Unlike other workspace-based probabilistic planners, WCO adapts its distribution dynamically according to the changes in the current roadmap. By efficiently combining information from both the workspace and the configuration space, WCO is less prone to misleading workspace information, even though it does not eliminate this problem entirely. Furthermore, by dynamically adapting its sampling distribution, WCO does not waste resources for oversampling subsets of $\mathcal{F}$ that have been well-represented by the current roadmap. In addition, unlike most workspace-based probabilistic planners, WCO uses multiple feature points. It generates a sampling distribution from each feature point of the robot and then combines them with adaptive hybrid sampling approach. By doing so, WCO implicitly incorporates more information about the robot's kinematics constraints in generating its sampling distribution. This further improve the ability of WCO in handling misleading workspace information.

Unlike other adaptive probabilistic planners, WCO uses workspace information along with the current state of the roadmap and the sampling history, instead of sampling history alone, to adapt its sampling distribution. By doing so, WCO uses less number of samples to learn which regions of $\mathcal{F}$ is more likely to improve the quality of the current roadmap. Hence, it is able to be more responsive to the changes in the roadmap.

Nevertheless, WCO still has difficulties when workspace information does not give any useful information. One example is when the projection of the narrow passages region of $\mathcal{F}$ lie in a single workspace triangle. One possible remedy for this is to use a more sophisticated strategy for sampling a configuration, given a point in the workspace. For instance, instead of sampling a configuration uniformly at random from the lift mapping of the given workspace point, we may use Randomized Bridge Builder [Hsu et al., 2005] which is specifically designed to sample narrow passages.

Another issue in WCO is choosing the right set of feature points. Currently we only have a heuristic for choosing this set of points. A better way of choosing the feature points may lead to better performance of WCO. Furthermore, it would be interesting to use a more sophisticated robot's feature. For instance, instead of representing a robot as a set of points, we may try to use a set of edges or faces of the robot. The main issue is how to efficiently incorporate these more sophisticated robot features to the planner.

Chapter 5

Understanding Adaptive Hybrid Sampling

Although adaptive hybrid sampling (AHS) has been introduced in [Hsu et al., 2005] and we have used it to combine the component samplers of WCO, its behaviour has not been fully explored. In this chapter, we further explore the behaviour of AHS. AHS is an adaptive sampling strategy that combines multiple sampling strategies, based on their sampling history. It assigns a weight to each strategy and dynamically updates the weights according to the performance of each strategy. Furthermore, AHS is cost-sensitive in the sense that the weight assigned to a sampler is normalized by the cost used by the sampler to sample a configuration. In the current WCO, we do not need to incorporate cost because in general all of its component samplers use roughly the same cost. However if we want to slightly modify WCO, for instance by replacing the uniform component sampler with a more sophisticated sampler for covering the free-space, we do need to take the cost into consideration. To understand AHS better, we first put AHS in a reinforcement learning framework. We then empirically show the importance of a suitable cost function and propose a more suitable cost function. Our experimental results show that as the narrow passages problem becomes more severe, AHS with the new cost function becomes significantly faster than AHS with the original cost function.

We start by presenting an overview of the AHS approach in Section 5.1. We

then present related works in reinforcement learning that becomes the basis of AHS in Section 5.2. Next, in Section 5.3, we put PRM planning with AHS in a reinforcement learning framework. Then, we present our new cost function along with the experimental setup and results in Section 5.4. Finally, we discuss possible future work on AHS in Section 5.5.

5.1 Overview of Adaptive Hybrid Sampling

Adaptive Hybrid Sampling (AHS) is an adaptive sampling strategy that combines multiple component samplers using a simple reinforcement learning method. The idea is to use reinforcement learning, based on sampling history, to infer which component sampler is more suitable for sampling the robot's free-space. AHS will then use the better component sampler more often. It assigns a weight to each component sampler, where higher weight is given to component samplers that have been performing well in the past. To measure the performance of a component sampler, AHS uses a reward and cost function. A reward is given to a component sampler whenever it samples a milestone that improves the coverage and/or connectivity of the current roadmap. While the cost is the time used for sampling the milestone. Obviously, a component sampler with larger total reward per unit cost is considered as a better component sampler. To sample a configuration, AHS chooses which component sampler to run based on the assigned weights. The overall algorithm is shown in Algorithm 5.1. It is based on the standard multi-query PRM approach as described in Section 2.2. The details of the primitive `FreeConf` for checking whether a configuration is collision free or not and the primitive `AllSolved`() for checking whether all queries have been solved or not are presented in Appendix A.

Notice that we slightly modify AHS in [Hsu et al., 2005] such that it can choose a different component sampler after a configuration is sampled, regardless of whether the configuration is collision-free or not. By doing so, the planner avoids getting stuck at one of the component sampler that may require a huge amount of samples before a collision-free configuration is sampled. Since in our modified

Algorithm 5.1 AHS(Q, N)

1: Let $p_{S_j}(s_i)$ be the probability of picking a component sampler s_i at iteration-j. Initialize $p_{S_0}(s_i) = 1/L$, for $i = 1, 2, \ldots, L$.
2: Set $j = 0$.
3: **repeat**
4: Pick a component sampler s_i from $\mathcal{S} = \{s_1, \ldots, s_L\}$ with probability p_{S_j}.
5: Sample a configuration q using s_i.
6: **if** `FreeConf`(q) returns true **then**
7: Insert q to the roadmap $\mathcal{R}$.
8: Increment j.
9: Compute the probability for the next iteration $p_{S_j}(s_i)$ for $i = 1, 2, \ldots, L$.
10: **until** `AllSolved`(Q, $\mathcal{R}$, Ψ) is true or $\mathcal{R}$ contains N milestones.
11: Return Ψ.

version, an action refers to sampling a configuration, instead of sampling a collision-free configuration, our modified version normalizes the probability by the cost for getting a configuration, instead of getting a collision-free configuration used in the original AHS. Due to this normalization, assuming that the number of samples for generating a milestone at different time are similar, the expected number of samples generated by each component sampler using our modified version is the same as that of the original AHS.

The probability $p_{S_j}(s_i)$ of choosing component sampler s_i at time j is updated using the following function:

$$p_{S_j}(s_i) = \frac{p^*_{S_j}(s_i)/c_i}{\sum_{k=1}^{L} p^*_{S_j}(s_k)/c_k} \tag{5.1}$$

where c_i is the estimated cost of running component sampler s_i to sample a configuration. To estimate the cost c_i, we use the number of collision checks needed to sample and insert the most recent milestone generated by s_i, averaged over the number of samples to generate the most recent milestone. We will discuss more about cost estimation in Section 5.4.1. The probability $p^*_{S_j}(s_i)$ is the cost-insensitive probability which is computed as:

$$p^*_{S_j}(s_i) = (1-\gamma)\frac{w_i(j)}{\sum_{k=1}^{L} w_k(j)} + \frac{\gamma}{L}, \qquad i = 1, \ldots, L, \tag{5.2}$$

where $w_i(j)$ is the weight of s_i in time j and $\gamma \in [0, 1]$ is a fixed constant. The

cost-insensitive probability consists of two parts. The first part is the weighted sum of the component samplers. The weight is based on the estimated cumulative reward of the action. It gives higher value to component samplers that have been performing well. How the weight and reward are computed is discussed below. The second part is the same for all component samplers. Its purpose is to ensure that all component samplers have a chance to show its performance.

The weight of a sampler s_i at time $j+1$ is computed as follows:

$$w_i(j+1) = w_i(j) \exp\left((r/p_{S_j}(s_i))\gamma/L\right). \tag{5.3}$$

where $w_i(0) = 1$ for each component sampler s_i and r is the reward given to s_i at time j. The reward is a heuristic to measure the usefulness of s_i in improving the coverage and connectivity of the current roadmap $\mathcal{R}$. A reward of one is given whenever the number of connected components of $\mathcal{R}$ changes. Increment of the number of components of $\mathcal{R}$ means that the new milestone can not be connected to any other components already in $\mathcal{R}$ and hence indicates that s_i improves the coverage of $\mathcal{R}$. Decrement of the number of components of $\mathcal{R}$ means that two or more components of $\mathcal{R}$ becomes connected due to the new milestone, and hence indicates that s_i improves the connectivity of $\mathcal{R}$. In other cases, a zero reward is given. Furthermore, the reward is divided by the probability $p_{S_j}(s_i)$ of choosing s_i at time j. It makes the weight of a component sampler increases more when $p_{S_j}(s_i)$ is low and less when $p_{S_j}(s_i)$ is high. As a result, the planner is responsive to the changes in the performance of a sampler, but mere luck will not cause a sampler to be favored much.

Under suitable assumptions, it has been shown that AHS is competitive against the best component sampler [Hsu et al., 2005]. More precisely, the following competitive ratio holds:

$$R_{\max} - R \leq (\mathrm{e}-1)\gamma R_{\max} + \frac{L \ln L}{\gamma}, \tag{5.4}$$

where R is the expected total reward received by AHS and $R_{\max}$ is the total reward received by the best component sampler if it is always chosen to run. This means

that AHS performs almost as well as the best component sampler, without knowing in advance which component sampler is the best.

5.2 Related Works

AHS is based on the simplest reinforcement learning problem, called the multi-armed bandit problem. Below, we will briefly review reinforcement learning and multi-armed bandit problem.

Reinforcement learning concerns with learning how to act by interacting with the environment. In general, a reinforcement learning agent may perform a set of actions in the environment and each of these actions may yield different reward. The agent will then try to discover the action or sequence of actions that will yield the maximum reward by trying the actions and learning from the reward it receives. A more elaborate explanation can be seen in [Sutton and Barto, 1998].

Now, back to our path planning problem. If we relax the problem of finding a suitable sampling distribution to finding a suitable sampling distribution among a set of potentially suitable sampling distributions, we can consider the planner as the agent and the potentially suitable sampling distributions as the actions. By assigning a reward function appropriately, the problem of finding a suitable sampling distribution among a set of potentially suitable sampling distributions can then be modeled as a reinforcement learning problem.

As in many reinforcement learning problem, the main difficulty in finding optimal action(s) is in balancing exploration and exploitation. The problem of balancing exploration and exploitation is often called as the multi-armed bandit problem [Sutton and Barto, 1998]. To get a lot of rewards, a learner needs to favor actions that generate more rewards. However to know which actions generate more rewards, the learner needs to try all actions. Hence, it needs to exploit what it already knows, but it also needs to explore in order to make better selection in the future [Sutton and Barto, 1998].

Many methods have been proposed for balancing exploration and exploitation. For example, ϵ-greedy [Sutton and Barto, 1998] always exploits the best action

greedily except for an ϵ fraction of time where it performs exploration by choosing an action uniformly at random. One drawback of ϵ-greedy is that aside from the best action, all other actions are considered equal. To alleviate this drawback, SoftMax [Sutton and Barto, 1998] uses Gibbs distribution to rank all actions from best to worst. Both of the above methods assume that the actions performed will not change the future reward. However, in our problem, the reward is determined by the position of the milestones in the current roadmap. Hence, an action may change the future reward. One of the few methods that does not assume that the actions performed will not change the future reward is EXP3 [Auer et al., 1995]. It uses γ fraction of the time for exploration and uses an exponential function to rank all actions from best to worse. All of these methods assume that all actions use the same cost.

In path planning, generally different component sampler uses different cost. AHS adapts EXP3 to incorporate cost. It normalizes the cost-insensitive probability, which is EXP3, with the cost spent by the component sampler.

5.3 Reinforcement Learning Framework for AHS

Although AHS uses a reinforcement learning method to find a good component sampler from a collection of component samplers, the reinforcement learning representation of the problem has not been articulated. In this section, we represent the problem of finding a suitable sampling distribution as a reinforcement learning problem.

A reinforcement learning problem consists of three main components, *i.e.*, state space, action space, and reward signal [Kaelbling et al., 1996]. To represent the problem of finding a suitable sampling distribution, we define a state as a 2-tuple $(\mathcal{R}, \eta)$ where $\mathcal{R}$ is a roadmap and $\eta \in [0, N_Q]$ is the number of queries that can be solved using the current roadmap, while N_Q is the number of the given queries. The initial state is $(\mathcal{R}_i, \eta_i) = (\mathcal{R}_0, 0)$, where $\mathcal{R}_0$ is a roadmap with zero milestone, before any milestone has been inserted. Assuming that all the given queries are solvable, the goal state is any state where $\eta = N_Q$. There can be many goal states,

but once the learner enters one of the goal states, it will only move to another goal state. The state space is then the set of all possible 2-tuples. The system changes from one state $(\mathcal{R}_j, \eta_j)$ to another $(\mathcal{R}_k, \eta_k)$, whenever $\mathcal{R}_k$ is the roadmap generated by inserting a sampled milestone q to $\mathcal{R}_j$.

An action constitutes of sampling a configuration $q \in \mathcal{C}$ using a particular component sampler and inserting q to the roadmap in the current state whenever q is collision-free. Different component sampler defines different action and uses different cost. Hence the action space may consists of all possible component samplers. However, for efficiency, AHS restricts its action space to a small number of component samplers.

In reinforcement learning, the reward signal is designed such that by maximizing the reward, the learner reaches the objective. In our case, the objective is to quickly reach one of the goal states. Since in general, a goal state is reached when the roadmap has adequately covers and captures the correct connectivity of $\mathcal{F}$, the reward signal is designed to favor sampling distributions that generate milestones that significantly improve the coverage and/or connectivity of the current roadmap. To do this, AHS uses the simplest reward signal, *i.e.*, a binary signal where one means good and zero means bad. It gives a reward of one whenever a sampling distribution, *i.e.*, a component sampler, samples a milestone that changes the number of connected component of the roadmap. And it gives zero reward, otherwise.

Using the above representation, the problem of finding a suitable sampling distribution can be considered as finding a sequence of actions that transforms the system from the initial state to one of the goal states, such that the total reward gathered over the total cost used is maximized. Since the learner does not know which action generates which reward, the learner needs to explore the action space to find which action is more promising, and then exploit the action that generates the most reward. Assuming the reward does not change drastically between a short sequence of states, the learner can predict which actions generate more rewards by trying them. Furthermore, to maximize the total reward over total cost, the learner needs to normalize its prediction with the cost for performing

the action. Since the cost of an action may be a priori unknown, the learner estimates the cost by trying them. In short, the problem of finding a suitable sampling distribution is transformed to the problem of balancing exploration and exploitation in reinforcement learning.

5.4 Experimental Setup and Results

In this section, we present the results of our empirical study on the behaviour of AHS. The results in [Hsu et al., 2005] show that AHS performs close to the performance of the best component sampler. In this section, we present further exploration on the behaviour of AHS. First, we propose a way to choose the cost function and explore the effect of different cost functions to AHS's performance. Our results show that cost function is critical to the overall performance of AHS. As the narrow passage problem becomes more severe, AHS with our new cost performs significantly faster than AHS with the original cost. Then, we explore the effect of the parameter γ to AHS's performance. Our results indicate that AHS is not sensitive to this parameter, which strengthen the user friendliness of AHS and strengthen the importance of a good cost function.

We implemented AHS using C/C++. Implementation details of the primitives, *i.e.*, `FreeConf`, `FreePath`, and `AllSolved` are presented in Appendix A. All experiments were conducted on a PC with a 3 GHz Intel Pentium 4 and 1 GB memory. All the experiments in this chapter uses 11 component samplers, *i.e.*, 1 uniform sampler, 5 RBB with parameters: 0.1, 0.3, 0.5, 0.7, 0.9, and 5 Gaussian sampler with parameters: 0.1, 0.3, 0.5, 0.7, 0.9.

5.4.1 Cost Function

As discussed in Section 5.3, the problem of finding a suitable sampling distribution can be rephrased as maximizing the total reward gathered over the total cost used. AHS tries to achieve this by first computing a cost-insensitive probability. This probability represents a prediction on the reward that can be gained by performing a particular action. Then, AHS normalizes the cost-insensitive probability with

the estimated cost for performing the action.

The cost function should estimate the time taken by a component sampler to sample a configuration and insert the new configuration to the roadmap whenever it is collision-free (Algorithm 5.1 line 5–7). Since not every sampled configuration is collision-free, we compute the cost of an action as the average cost over the number of samples for generating a collision-free configuration. More precisely, $c_i = \frac{cm_i}{N_{si}}$, where cm_i is the cost for sampling a collision-free configuration and inserting it to the current roadmap, while N_{si} is the number of samples for generating a collision-free configuration. We assign N_{si} as the number of samples needed by s_i to sample the previous milestone.

In [Hsu et al., 2005], the cost cm_i is computed as the number of collision checks for sampling and inserting the previous milestone to the roadmap. When the number of collision checks dominates the total cost, this estimation performs well. However, with a better connection strategy, one can keep the number of collision checks low, for instance by trying to link the new milestone only to other milestones that lie in different connected components of the roadmap. As a result, when the number of milestones in the roadmap is large, the time for performing collision checks no longer dominates the time for sampling and inserting a new milestone to the roadmap. Instead, the time for computing kNN (part of the operation in Algorithm 5.1 line 7) becomes comparable and may even dominate the time for sampling and inserting a new milestone to the roadmap. The reason is that kNN computation may take $O(N_m)$ time for answering a query, where N_m is the number of milestones in the current roadmap. Current nearest neighbor algorithm that uses near-linear storage would take $\min(2^{O(n)}, nN_m)$ query time, where n is the number of dimensions of $\mathcal{F}$. And in practice, this sophisticated nearest neighbor search exhibits linear query time for n between 10 and 20 [Indyk, 2004]. Hence, as the path planning problem becomes more complicated and more milestones are needed to solve the problem, the number of milestones in the current roadmap may significantly affect the time used by an action.

Therefore, we propose to assign cm_i based on the number of milestones in the current roadmap and the number of collision checks for sampling and inserting the

previous milestone to the roadmap. To find the cost function cm, we use multiple linear regression model [Montgomery et al., 2001], $cm = c_1 \cdot cc + c_2 \cdot N_m$ where cc is the number of collision checks used to sample and insert the previous milestone to the current roadmap, N_m is the number of milestones in the current roadmap, and c_1 and c_2 are constants. To find the constants, we need data points and an estimator. To get the data points, we ran experiments with the environment and robot in Figure 5.1. We ran the basic-PRM and Gaussian strategy for 30 times each until the roadmap contains $10,000$ milestones, and capture the time for sampling a milestone and inserting the milestone to the current roadmap. So, we have data points for N_m between 1 and $10,000$ for each planner. For each value of N_m, the number of collision checks cc and the time cm are the average of the 30 runs of the planner. For each planner and scenario, the constants are then estimated using least-squares estimation, *i.e.*,

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \approx \left(\begin{bmatrix} CC & NM \end{bmatrix}^T \cdot \begin{bmatrix} CC & NM \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} CC & NM \end{bmatrix}^T \cdot CM \quad (5.5)$$

where $[\cdot]^T$ is the transpose matrix, CC, NM, and CM are column vectors of $10,000$ elements, where each element of the vectors represents the number of collision checks cc, the number of milestones N_m, and the time cm for a data point. From computing (5.5) for the different planners we use, we set $cm = 250 \cdot cc + N_m$.

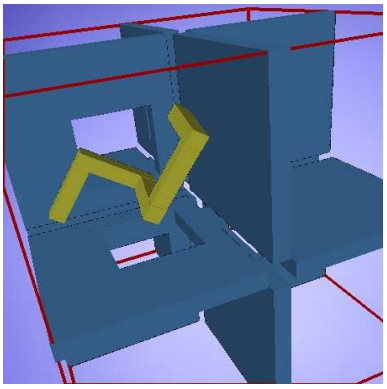


Figure 5.1: Scenario for finding a cost function. The robot is a 6-dofs rigid-body robot.

Notice that with this new cost function, when the number of milestones in the roadmap dominates the number of collision checks needed by each component sampler s_i, $i = 1, \ldots, L$ to sample and insert a milestone to the current roadmap, the cost cm_i is the same for all component samplers. Recall from Section 5.1 that under suitable assumption, the expected number of samples and hence the expected number of milestones generated by each component sampler of our modified AHS is the same as those generated by the original AHS,

which allows the planner to switch to another sampler only when a collision-free configuration has been sampled. Now since when cm_i of all component samplers are the same, the original AHS favors component samplers that generate more reward per milestone, so does with our modified AHS when cm_i is the same for all component samplers. This is reasonable as in this case, the most time consuming part of sampling and inserting a milestone is now the kNN computation, which depends on the number of milestones in the roadmap.

To test the effect of modifying the cost function, we performed experiments on two sets of scenarios, shown in the left part of Figure 5.2. Each set consists of three scenarios, where the size of the narrow passages are varied. We ran AHS with the original cost function and with our proposed cost function for 30 times each, in each scenario. The average results are shown in the right part of Figure 5.2.

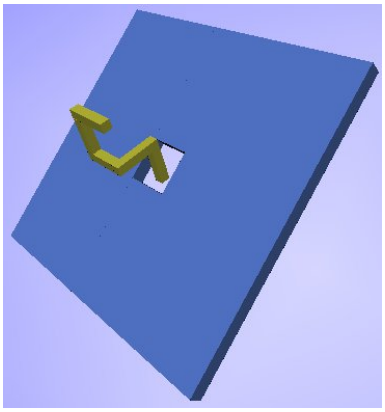
As expected, the results show that as the number of milestones increases, AHS with the new cost function performs significantly better than AHS with the original cost function. When the number of milestones is small, the performance of the two strategies are comparable, because the new cost is dominated by the number of collision checks and hence the old and new cost are similar. As the number of milestones increases, the time spent for kNN computation is comparable and may even dominate the time spent for collision check computation. Hence, the new cost function estimates the actual cost better than the original cost function. As a result, AHS with the new cost function can better identify the sampler that generates more total reward per unit cost, and converges to a good roadmap faster.

5.4.2 Parameter γ

Although AHS uses only 1 parameter, it would be useful to know how sensitive AHS is to this parameter. If AHS is not sensitive to this parameter, then the user can arbitrarily set γ without degrading the performance of AHS significantly.

We tested AHS on the scenarios shown in Figure 5.3. Each scenario was run with AHS using various γ. Each scenario and each AHS with a particular γ value was run 30 times. The average of the runs are shown in Table 5.1.

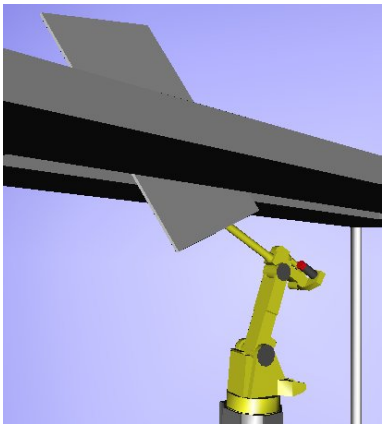
The results show that AHS's performance is not sensitive to γ. In [Hsu et al.,



Test 1

AHS with the original cost function			
Hole size	T_{tot} (sec)	N_{mil}	N_{CC}
$0.23 \times 0.23 \times 0.07$	49	6,193	195,273
$0.21 \times 0.21 \times 0.07$	192	16,698	536,822
$0.19 \times 0.19 \times 0.07$	1,274	50,355	1,628,927
AHS with the proposed cost function			
Hole size	T_{tot} (sec)	N_{mil}	N_{CC}
$0.23 \times 0.23 \times 0.07$	36	4,560	171,241
$0.21 \times 0.21 \times 0.07$	178	11,761	516,808
$0.19 \times 0.19 \times 0.07$	480	25,705	1,262,972

Results



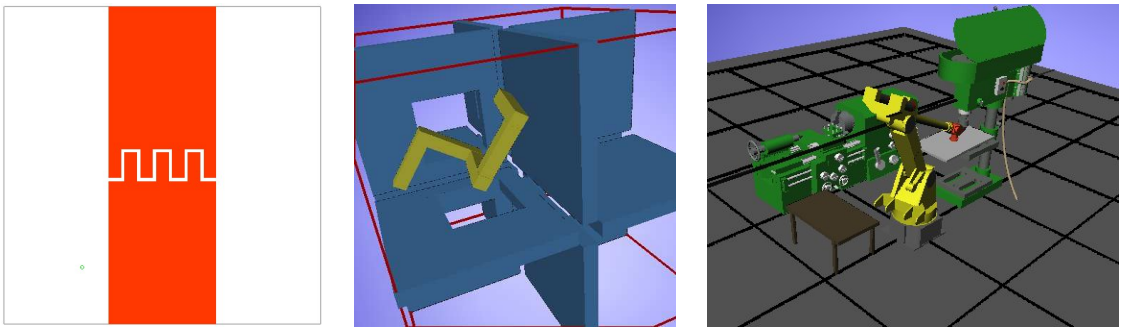
Test 2

AHS with the original cost function			
Slot width	T_{tot} (sec)	N_{mil}	N_{CC}
0.23	30	2,754	114,276
0.20	66	6,809	287,067
0.17	2,458	75,021	3,320,156
AHS with the proposed cost function			
Slot width	T_{tot} (sec)	N_{mil}	N_{CC}
0.23	32	2,936	121,543
0.20	66	6,606	281,822
0.17	1,284	52,974	2,418,580

Results

Figure 5.2: Scenarios and results of testing the cost function. T_{tot} is the total running time. N_{mil} is the number of milestones required for answering the query. N_{CC} is the number of collision checks required for answering the query. **Test 1**: A 6-dofs rigid-body robot must pass through the narrow opening in order to answer the given query. This test consists of 3 scenarios, in which the size of the narrow opening is varied. The figure shows the scenario with the smallest opening. **Test 2**: A 6-dofs robot manipulator with its end-effector holding a large plate needs to maneuver through a narrow slot. This test consists of 3 scenarios, in which the width of the narrow slot is varied. The figure shows the scenario with the smallest slot. The second scenario with slot width = 0.20 is the same as the one used as **Test 4** in Chapter 3 and Chapter 4.

2005], it has been argued that AHS takes the burden of "tweaking" the component samplers's parameters from the user. Our results further strengthen the argument of the user friendliness of AHS. Practically, AHS frees the user from the burden of any parameter "tweaking".



Test 3 Test 4 Test 5

Figure 5.3: Test scenarios for testing the effect of γ. **Test 1**: A point robot moves from the lower left corner to the lower right corner by passing through the narrow opening. **Test 2**: A 6-dofs rigid-body robot must pass through 6 out of 7 narrow openings in order to answer the given query. This test is the same as Test 3 used in Chapter 3 and Chapter 4. **Test 4**: A 6-dofs robot manipulator needs to move the small thin plate to the narrow opening of the machine.

Table 5.1: The effect of γ on the overall performance of AHS.

Scenario	T_{tot} (sec)					
	AHS					Uniform weight
	$\gamma = 0.1$	$\gamma = 0.3$	$\gamma = 0.5$	$\gamma = 0.7$	$\gamma = 0.9$	
Test 3	43.85	44.86	49.18	44.07	57.81	93.33
Test 4	60.48	50.93	53.26	50.08	63.68	97.44
Test 5	0.48	0.47	0.45	0.47	0.41	0.65

Furthermore, the insensitivity of AHS to γ strengthen our argument on the importance of cost to AHS strategy. Observation on (5.1) and (5.2) indicates that when the overall performance of AHS does not vary much due to γ, the weight w_i for each sampler $i = 1, \ldots, L$ is close to $\frac{1}{L}$. Assuming that the reward is a good feedback for constructing a good roadmap and that all component samplers can sample important regions of $\mathcal{F}$, in general the total reward gathered by different component samplers when run alone to construct a good roadmap should roughly be the same. Now, the weight w is based on the estimated cumulative reward [Auer et al., 2002]. Hence, assuming that the estimated cumulative reward estimates the total reward well, after enough learning time, the estimated cumulative reward of all component samplers are almost the same and hence w of all component samplers are almost the same too. Since w of each component sampler is almost the same,

the probability $p_{S_j}(s_i)$ highly depends on the cost c_i. Hence a good estimation of the cost is critical to the overall performance of AHS. Note however that this does not imply that the sophisticated weight function in (5.3) is utterly useless for AHS strategy. When some of the component samplers can not sample some important regions of $\mathcal{F}$, then the total reward that can be gathered by these samplers can be much lower than other samplers. In this case, the weight computation in (5.3) enables AHS to disfavor these component samplers.

5.5 Discussion

In this chapter, we have further explored the behaviour of adaptive hybrid sampling (AHS) approach which is used to combine the component samplers of WCO (Chapter 4). We put AHS in a reinforcement learning framework. We strengthen the user friendliness of AHS by showing that it is insensitive to the variation of its only parameter γ. Then, we empirically show the importance of cost function and propose a new cost function. Our experimental results show that as the narrow passages problem becomes more severe, AHS with the new cost function becomes significantly faster than AHS with the original cost function.

Currently, AHS tries to find the best component sampler. However, many sampling strategies work best when they are combined with another strategy. For instance, combining RBB and uniform strategy is better than RBB or uniform alone. Most free-space consists of both narrow passages and wide-open spaces. So for the planner to perform well, the roadmap constructed must represent both regions of the free-space well. Therefore, it is useful to explore ways to combine the samplers such that the combination is close to the best combination of the component samplers.

AHS is not restricted to motion planning problems. When many heuristics with different strength have been proposed for solving a problem, combining these heuristics with AHS strategy may be useful. In general, different heuristics perform well in different instances of the problem. Using AHS to combine the different heuristics ensures that the overall performance will be close to the performance

of the best heuristic for any instance of the problem. In short, AHS takes the problem of choosing a suitable heuristic off the user. However, of course for AHS to perform well, we need a good reward and cost estimation function.

Chapter 6

Conclusion

Despite the hardness of high dimensional path planning problems, probabilistic path planning have solved many high dimensional problems, expanding the use of path planning beyond robotics, to areas as diverse as computer graphics [Koga et al., 1994], computer aided-design [Chang and Li, 1995], and computational biology [LaValle et al., 2000, Amato et al., 2002, Apaydin et al., 2003]. However, despite the experimental success of probabilistic path planning, its performance degrades significantly when the configuration space contains narrow passages. And unfortunately, as the dimensionality of the problem increases, the narrow passages tend to become narrower, and hence worsen the performance of probabilistic path planning.

This thesis shows that workspace provides useful information for generating a suitable sampling distribution, which then improves the performance of probabilistic path planning in solving narrow passages problem significantly. We start by articulating the distinction between two main components of random sampling, *i.e.*, sampling distribution and sampling source, in probabilistic path planning. This articulation enables us to empirically show the importance of a suitable sampling distribution to the overall performance of the planner. Furthermore, our empirical study indicates that suitable sampling distributions depend on the visibility property of the robot's free-space. The low dimensionality and explicit representation of the workspace along with the close relation between obstacles in the workspace and forbidden regions in the configuration space have encourage us to explore the

use of workspace information in generating suitable sampling distributions. Our exploration reveals that the visibility sets of points in the robot's free-space are strongly related to the visibility sets of points in the workspace free-space. Moreover, even a simple strategy for utilizing workspace information turns out to show comparable performance to recent probabilistic path planner. These results encourage us to exploit workspace information further in our new probabilistic path planner WCO. Our analysis shows that WCO is probabilistically complete and under certain conditions, the failure probability of WCO is lower than that of the basic-PRM. And our experiments show that WCO performs up to 28 times faster than the basic-PRM and up to 8 times faster than recent probabilistic path planner that has been shown to perform well. Furthermore, as the dimensionality of the configuration space increases, the performance of our new planner decreases slower than other recent probabilistic path planner. We have also successfully implemented WCO in a simulated bridge inspection scenario involving a 35-dofs robot.

6.1 Summary of Contribution

This thesis shows that workspace provides useful information for generating a suitable sampling distribution for probabilistic path planning in solving narrow passages problem.

We performed an empirical study to understand the role and importance of sampling distribution in probabilistic path planning, and to find possible avenues for alleviating narrow passages problem. We start by articulating the distinction between sampling distribution and sampling source, which has been blurred in the probabilistic path planning literature. This articulation enable us to identify and empirically show that sampling distribution is critical to the success of probabilistic path planning, because it relates to the uncertainty that arises from our lack of information about the shape of the robot's configuration space. Our empirical study indicates that a suitable sampling distribution depends on the visibility property of the robot's free-space. The low dimensionality and explicit representation of the

workspace along with the close relation between obstacles in the workspace and forbidden regions in the configuration space suggest that workspace information is potentially useful for estimating the visibility property of the robot's free-space. Furthermore, although the shape of the robot's free-space is unknown, partial information about the free-space can be gathered as the roadmap is being constructed and the sampling distribution can be adjusted accordingly. Therefore, another potential strategy for generating a suitable sampling distribution is by adjusting the sampling distribution dynamically over time. In short, our study suggests that utilizing workspace information and dynamically adapting sampling distribution over time are two potential strategies for improving current probabilistic path planners.

Further exploration on the relation between the configuration space and the workspace shows that although the two spaces are different, their visibility property are related. Under certain conditions that *do not* depend on the dimensionality of the configuration space, the visibility set of a configuration in the free-space is a subset of the lift mapping of the visibility set of the configuration's projection in workspace free-space. This implies that to find configurations that can be directly connected to a configuration q of the free-space, it is safe to restrict our search space to those configurations that place the robot's feature point at the visibility set of the projection of q in the workspace free-space $\mathcal{W}_{\mathcal{F}}$. Moreover, the volume of the visibility sets in the free-space is bounded by a constant multiplication of the volume of the visibility sets in the workspace free-space. Since the number of milestones for solving a path planning problem within a certain probability of success can be bounded in terms of the volume of the visibility set of points in the free-space, our result indicates that the number of milestones can also be bounded in terms of the volume of the visibility set of points in the workspace free-space. These results imply that the workspace contains useful information about the visibility property of the robot's free-space. And based on our empirical study, these results indicate that workspace provides useful information for generating suitable sampling distributions for probabilistic path planning.

The question remains as to how workspace information can be used to efficiently generate a suitable sampling distribution over the configuration space. As our first

attempt to address this issue, we present a simple workspace-based path planner, called WIS. WIS uses the width of passages in the workspace to estimate the size of the visibility set of subsets of the robot's free-space and constructs a static sampling distribution based on this estimation. Our experimental results show that this simple sampling strategy is comparable to recent probabilistic path planner that has been shown to perform well. Our analysis shows that the failure probability of WIS converges to zero exponentially in the number of milestones, provided a solution exists. Furthermore under certain conditions which often happen when the robot moves inside the narrow passages of the workspace, the upper bound of the failure probability of WIS is lower than that of the basic-PRM.

Encouraged by the above results, we propose a workspace-based path planner, called WCO, that uses workspace information to generate *dynamic* sampling distribution. It combines workspace information with sampling history and the current state of the roadmap to dynamically adapt its sampling distribution. Unlike previous probabilistic path planners that generate dynamic sampling distribution, WCO uses both workspace information and sampling history instead of sampling history alone. By doing so, WCO blends workspace information that provides a rough global information about the configuration space with sampling history that provides a more detailed local information of the regions around the sampled configurations. Furthermore, WCO consists of multiple component samplers, where each sampler is based on a point of the robot. This implies that WCO takes more robot information into consideration. The two main traits above enable WCO to perform significantly faster than other recent probabilistic path planner. Our analysis shows that the failure probability of WCO converges to zero exponentially in the number of milestones, provided a solution exists. Furthermore, when its estimation satisfies a certain criteria, the upper bound on the failure probability of WCO is lower than that of the basic-PRM. Our experimental results show that WCO performs up to 28 times faster than the basic-PRM. Furthermore, as the dimensionality of the robot's configuration space increases, WCO's performance decreases slower than other recent probabilistic path planner. We have also successfully implemented WCO in a simulated bridge inspection scenario involving a

35-dofs robot.

Next, we further explore the behaviour of the adaptive hybrid sampling (AHS) approach [Hsu et al., 2005], a strategy used by WCO to combine its component samplers. We empirically show the importance of cost function and propose a new cost function. Our experimental results show that as the narrow passages problem becomes more severe, AHS with the new cost function becomes significantly faster than AHS with the original cost function.

6.2 Future Work

Although this thesis focuses mainly on multi-query probabilistic path planning, workspace information is likely to be useful in single-query probabilistic path planning, too. The goal of single-query path planning is to solve a given query as fast as possible. Therefore, it is desirable that the planner does not waste time for exploring regions of the configuration space that will not contribute to the final path. However, works on single-query probabilistic planners [Hsu et al., 1999, Kuffner and LaValle, 2000, Sánchez and Latombe, 2001] have not thoroughly explored ways of reducing the exploration of configuration space regions that are unlikely to contribute to the final path. WCO indicates that workspace information can be used to identify regions that are more likely to contribute to the final path and hence help in generating a more suitable sampling distribution. Nevertheless, more research is needed to find an effective method for combining multiple samplers for single-query planning.

An open problem that has not received much attention in probabilistic path planning is how to a priori determine the number of milestones needed by a planner to reach a certain percentage of success. Currently, the burden of choosing an appropriate number of milestones rests with the user. Too little milestones causes many unsolved queries, while too many milestones requires significantly more time for building the roadmap. It is true that several works on the analysis of the basic-PRM [Kavraki et al., 1995, Kavraki et al., 1996a, Švestka, 1996, Hsu et al., 1999, Ladd and Kavraki, 2004] have provided a bound on the necessary

number of milestones. However, these bounds are either based on the property of a solution path [Kavraki et al., 1995, Ladd and Kavraki, 2004], which is unknown before the solution is found. Or they are based on the geometric property of the high dimensional free-space [Kavraki et al., 1996a, Švestka, 1996, Hsu et al., 1999]. For instance, the analysis in [Kavraki et al., 1995], provides a bound based on the volume of the smallest visibility set in the free-space. Since the free-space is unknown, computing this property exactly is infeasible. Moreover, computing a reliable estimation using Monte Carlo is likely to take as much time as building a good roadmap [Hsu et al., 1999]. Hence, despite these theoretical bounds, the number of milestones that should be sampled to reach a certain success rate remains incomputable. In this thesis, we have shown that under certain conditions, the size of the visibility sets in the free-space is upper bounded by a constant multiplication of the visibility sets in the workspace free-space. Furthermore, from the proof in Chapter 3, this constant can be computed from the robot's kinematics information. This result suggests that we may use the volume of the largest visibility set of a point in the workspace free-space and then use the theoretical bound in [Kavraki et al., 1995] to a priori compute a loose lower bound on the number of milestones needed for the planner to reach a certain success rate. In this direction, the main issue is how to efficiently compute the size of the visibility sets in the workspace free-space. Furthermore, finding a tighter bound could be useful. A different direction may be to explore efficient method to directly estimate the number of required milestones using workspace information.

In this thesis, we focus mainly on finding geometric paths and ignore all differential constraints, such as non holonomic and dynamic constraints of the robot's motion. Taking differential constraints into consideration means that the planner needs to find not just a sequence of configurations, but also a sequence of velocity and/or robot's control. Hence, planning is performed in the robot's state space, where a state is a tuple of configuration and tangent vector [Latombe, 1991]. Planning in the state space is in general harder than in the configuration space, as the dimensionality of the state space is significantly higher than the configuration space. Our experimental results show that workspace-based sampling is effective

in slowing down the planner's performance degradation due to the increase of dimensionality. Hence, it may be useful to explore the use of workspace information when the robot's differential constraints are considered. In the WCO framework, one possibility is to adapt the workspace channel prediction of WCO such that the robot's velocity/control constraints are taken into account. One main issue would be how to map the tangent vector in the state space to and from the workspace. The simplest way is of course to just assume the tangent vector as free variables. However, this may not help much when the velocity/control contraints are very restricted. Thus, further research is needed to explore the use of workspace information in solving motion planning problems involving differential constraints.

With the advance of robotics technology, high dimensional robots, such as humanoid robots, start to enter our living room. When robots share a living space with humans, we can no longer assume that the environment is static and perfectly known. To be applicable, motion planners need to be responsive enough to the dynamic and uncertainty of the environment. These types of robots sense the changes in the workspace geometry using its sensors, and defines its motion in the high dimensional configuration space. Hence, the efficient mapping between the workspace and the configuration space, presented in this thesis, may be useful to handle the dynamic of the environment. Once the robot detects changes in the workspace geometry, the planner can quickly modify parts of the robot's path that have been invalidated. However, since the changes in the workspace geometry is uncertain due to sensor's noise, further research is needed to explore how uncertainty in the workspace information can be incorporated to help planning.

Bibliography

[Amato et al., 1998] Amato, N., Bayazit, O., Dale, L., Jones, C., and Vallejo, D. (1998). OBPRM: An obstacle-based PRM for 3D workspaces. In Agarwal, P. et al., editors, *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, pages 155–168. A. K. Peters, Wellesley, MA.

[Amato et al., 2002] Amato, N., Dill, K., and Song, G. (2002). Using motion planning to map protein folding landscapes and analyze folding kinetics of known native structures. *J. Computational Biology*, 10(2):239–255.

[Amenta et al., 2001] Amenta, N., Choi, S., and Kolluri., R. (2001). The power crust. In *Proc. ACM Symposium on Solid Modeling*, pages 249–266.

[Apaydin et al., 2002] Apaydin, M., Bruglag, D., Guestrin, C., Hsu, D., and Latombe, J.-C. (2002). Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. In *ACM International Conference on Research in Computational Biology (RECOMB)*, pages 12–21.

[Apaydin et al., 2003] Apaydin, M., Bruglag, D., Guestrin, C., Hsu, D., Latombe, J.-C., and Varma, C. (2003). Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. *J. Computational Biology*, 10(3–4):247–281.

[Auer et al., 1995] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. (1995). Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Annual Symposium on Foundations of Computer Science*, pages 322–331.

[Auer et al., 2002] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. (2002). The nonstochastic multiarmed bandit problem. *SIAM J. Computing*, 32(1):48–77.

[Barber et al., 1996] Barber, C., Dobkin, D., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hull. *ACM Transactions on Mathematical Software*, 22(4):469–483.

[Barraquand and Latombe, 1990] Barraquand, J. and Latombe, J.-C. (1990). A Monte-Carlo algorithm for path planning with many degrees of freedom. In *Proc. IEEE International Conference on Robotics & Automation*.

[Boor et al., 1999] Boor, V., Overmars, M., and van der Stappen, F. (1999). The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE International Conference on Robotics & Automation*, pages 1018–1023.

[Burns and Brock, 2005] Burns, B. and Brock, O. (2005). Toward optimal configuration space sampling. In *Proc. Robotics: Science and Systems*.

[Canny, 1988] Canny, J. (1988). *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA.

[Chang and Li, 1995] Chang, H. and Li, T.-Y. (1995). Assembly maintainability study with motion planning. In *Proc. IEEE International Conference on Robotics & Automation*, pages 1012–1019.

[Chaudhuri and Koltun, 2007] Chaudhuri, S. and Koltun, V. (2007). Smoothed analysis of probabilistic roadmaps. In *SIAM Workshop on Analytic Algorithms and Combinatorics*, page to appear, New Orleans, LA.

[Cheng et al., 2006] Cheng, H.-L., Hsu, D., Latombe, J.-C., and Sánchez-Ante, G. (2006). Multi-level free-space dilation for sampling narrow passages in PRM planning. In *Proc. IEEE International Conference on Robotics & Automation*, pages 1255–1260.

[Choset et al., 2005] Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., and Thrun, S. (2005). *Principles of Robot Motion : Theory, Algorithms, and Implementations*. The MIT Press.

[de Berg et al., 2000] de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (2000). *Computational Geometry Algorithms and Applications*. Springer Verlag, 2^{nd} edition.

[Foley et al., 1995] Foley, J., van Dam, A., Feiner, S., and Hughes, J. (1995). *Computer Graphics: Principles and Practice in C*. Addison-Wesley.

[Foskey et al., 2001] Foskey, M., Garber, M., Lin, M., and Manocha, D. (2001). A Voronoi-based hybrid motion planner. In *Proc. IEEE/RSJ International Conference on Intelligent Robots & Systems*.

[Fraichard, 1999] Fraichard, T. (1999). Trajectory planning in a dynamic workspace: a 'state-time' approach. *Advanced Robotics*, 13(1):75–94.

[Geraerts and Overmars, 2002] Geraerts, R. and Overmars, M. (2002). A comparative study of probabilistic roadmap planners. In Boissonnat, J. et al., editors, *Algorithmic Foundations of Robotics V—Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 43–59. Springer.

[Geraerts and Overmars, 2005] Geraerts, R. and Overmars, M. H. (2005). Reachability analysis of sampling based planners. In *Proc. IEEE International Conference on Robotics & Automation*, pages 406–412.

[Halperin et al., 1999] Halperin, D., Kavraki, L. E., and Latombe, J.-C. (1999). Robot algorithms. In Attalah, M., editor, *Algorithms and Theory of Computation Handbook*, chapter 21. CRC Press, Boca Raton, NY.

[Holleman and Kavraki, 2000] Holleman, C. and Kavraki, L. E. (2000). A framework for using the workspace medial axis in PRM planners. In *Proc. IEEE/RSJ International Conference on Intelligent Robots & Systems*.

[Hsu, 2000] Hsu, D. (2000). *Randomized Single-Query Motion Planning in Expansive Spaces*. PhD thesis, Dept. of Computer Science, Stanford University, Stanford, CA.

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

[Hsu et al., 2002] Hsu, D., Kindel, R., Latombe, J.-C., and Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255.

[Hsu et al., 1997] Hsu, D., Latombe, J.-C., and Motwani, R. (1997). Path planning in expansive configuration spaces. In *Proc. IEEE International Conference on Robotics & Automation*, pages 2719–2726.

[Hsu et al., 1999] Hsu, D., Latombe, J.-C., and Motwani, R. (1999). Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications (IJCGA)*, 9(4–5):495–512.

[Hsu et al., 2005] Hsu, D., Sánchez-Ante, G., and Sun, Z. (2005). Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE International Conference on Robotics & Automation*, pages 3885–3891.

[Indyk, 2004] Indyk, P. (2004). Nearest neighbors in high-dimensional spaces. In Goodman, J. E. and O'Rourke, J., editors, *Handbook of Discrete and Computational Geometry, chapter 39*. CRC Press. 2nd edition.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. P. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

[Kavraki et al., 1996a] Kavraki, L., Kolountzakis, M., and Latombe, J.-C. (1996a). Analysis of probabilistic roadmaps for path planning. In *Proc. IEEE International Conference on Robotics & Automation*, pages 3020–3025.

[Kavraki et al., 1995] Kavraki, L., Latombe, J.-C., Motwani, R., and Raghavan, P. (1995). Randomized query processing in robot path planning. In *Proc. ACM Symp. on Theory of Computing*, pages 353–362.

[Kavraki et al., 1996b] Kavraki, L., Švestka, P., Latombe, J.-C., and Overmars, M. (1996b). Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Transactions on Robotics & Automation*, 12(4):566–580.

[Knuth, 1998] Knuth, D. E. (1998). *The Art of Computer Programming*. Addison-Wesley, 2^{nd} edition.

[Koga et al., 1994] Koga, Y., Kondo, K., Kuffner, J. J., and Latombe, J.-C. (1994). Planning motions with intentions. In *Proceedings of SIGGRAPH 94*, pages 395–408.

[Kuffner and LaValle, 2000] Kuffner, J. J. and LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE International Conference on Robotics & Automation*, pages 995–1001.

[Ladd and Kavraki, 2004] Ladd, A. M. and Kavraki, L. E. (2004). Measure theoretic analysis of probabilistic path planning. *IEEE Transactions on Robotics & Automation*, 20(2):229–242.

[Latombe, 1991] Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers.

[Laumond, 1998] Laumond, J. (1998). *Robot Motion Planning and Control*, volume 229. Lectures Notes in Control and Information Sciences.

[LaValle et al., 2004] LaValle, S., Branicky, M., and Lindemann, S. (2004). On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research*, 23(7/8):673–692.

[LaValle et al., 2000] LaValle, S., Finn, P., Kavraki, L., and Latombe, J.-C. (2000). A randomized kinematics-based approach to pharmacophore-constrained conformational search and database screening. *J. Computational Chemistry*, 21(9):731–747.

[LaValle and Kuffner, 2001] LaValle, S. M. and Kuffner, J. J. (2001). Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400.

[Lin and Manocha, 2004] Lin, M. and Manocha, D. (2004). Collision and proximity queries. In Goodman, J. and O'Rourke, J., editors, *Handbook of Discrete and Computational Geometry*, chapter 35. CRC Press.

[Lindemann and LaValle, 2003] Lindemann, S. and LaValle, S. (2003). Incremental low-discrepancy lattice methods for motion planning. In *Proc. IEEE International Conference on Robotics & Automation*, pages 2920–2927.

[Montgomery et al., 2001] Montgomery, D. C., Peck, E. A., and Vining, G. G. (2001). *Introduction to Linear Regression Analysis*. Wiley-Interscience, 3rd edition.

[Morales et al., 2006] Morales, A. M. A., Pearce, R., and Amato, N. M. (2006). Metrics for analyzing the evolution of C-space models. In *Proc. IEEE International Conference on Robotics & Automation*, pages 1268–1273.

[Morales et al., 2004] Morales, M., Tapia, L., Pearce, R., Rodriguez, S., and Amato, N. M. (2004). A machine learning approach for feature-sensitive motion planning. In Erdmann, M. et al., editors, *Algorithmic Foundations of Robotics VI—Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 361–376. Springer.

[Moro, 1995] Moro, B. (1995). The full monte. *Risk*, 8(2).

[Motwani and Raghavan, 2000] Motwani, R. and Raghavan, P. (2000). *Randomized Algorithms*. Cambridge University Press.

[Niederreiter, 1992] Niederreiter, H. (1992). *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial & Applied Mathematics.

[Quinlan, 1994] Quinlan, S. (1994). Efficient distance computation between non-convex objects. In *Proc. IEEE International Conference on Robotics & Automation*, pages 3324–3329.

[Reif, 1979] Reif, J. H. (1979). Complexity of the mover's problem and generalization. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 421–427.

[Rodriguez et al., 2006] Rodriguez, S., Thomas, S., Pearce, R., and Amato, N. M. (2006). RESAMPL: A region-sensitive adaptive motion planner. In Akella, S. et al., editors, *Algorithmic Foundations of Robotics VII—Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*. Springer.

[Saha and Latombe, 2005] Saha, M. and Latombe, J.-C. (2005). Finding narrow passages with probabilistic roadmaps: The small step retraction method. In *Proc. IEEE/RSJ International Conference on Intelligent Robots & Systems*.

[Sánchez and Latombe, 2001] Sánchez, G. and Latombe, J.-C. (2001). A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Proc. International Symposium on Robotics Research*.

[Shoemake, 1985] Shoemake, K. (1985). Animating rotation with quaternion curves. *ACM SIGGRAPH*, 19(3):245–254.

[Siméon et al., 2000] Siméon, T., Laumond, J., and Nissoux, C. (2000). Visibility-based probabilistic roadmaps for motion planning. *J. Advanced Robotics*, 14(6):477–494.

[StatPac, 1997] StatPac (1997). Survey sampling methods. `http://www.statpac.com/surveys/sampling.htm`.

[Sun et al., 2005] Sun, Z., Hsu, D., Jiang, T., Kurniawati, H., and Reif, J. (2005). Narrow passage sampling for probabilistic roadmap planners. *IEEE Trans. on Robotics*, 21(6):1105–1115.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

[Švestka, 1996] Švestka, P. (1996). On probabilistic completeness and expected complexity for probabilistic path planning. Technical Report UU-CS-1996-08, Utrecht University, Department of Information & Computing Sciences, Utrecht, the Netherlands.

[Thrun et al., 2000] Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Hähnel, D., Rosenberg, C., Roy, N., Schulte, J., and Schulz, D. (2000). Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 19(11):972–999.

[Thrun et al., 2004] Thrun, S., Thayer, S., Whittaker, W., Baker, C., Burgard, W., Ferguson, D., Hähnel, D., Montemerlo, M., Morris, A., Omohundro, Z., Reverte, C., and Whittaker, W. (2004). Autonomous exploration and mapping of abandoned mines. *IEEE Robotics and Automation Magazine*, 11(4):79–91.

[Toussaint et al., 1993] Toussaint, G. T., Verbrugge, C., Wang, C., and Zhu, B. (1993). Tetrahedralization of simple and non-simple polyhedra. In *Proc. 5th Canadian Conference on Computational Geometry*, pages 24–29.

[van den Berg and Overmars, 2005] van den Berg, J. P. and Overmars, M. H. (2005). Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. *International Journal of Robotics Research*, 24(12):1055–1072.

[Wilmarth et al., 1999] Wilmarth, S., Amato, N., and Stiller, P. (1999). MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE International Conference on Robotics & Automation*, pages 1024–1031.

[Yang and Brock, 2004] Yang, Y. and Brock, O. (2004). Adapting the sampling distribution in PRM planners based on an approximated medial axis. In *Proc. IEEE International Conference on Robotics & Automation*.

Appendix A

Primitives

A.1 Implementation of `FreeConf`

We use LEDA library to implement `FreeConf` for robots working in 2D workspace and we use Quinlan's distance computation method [Quinlan, 1994] as `FreeConf` for robots working in 3D workspace. The idea of Quinlan's method is to build a hierarchical representation of the objects a-priori, such that it can later compute the (exact or approximate) distance between two objects fast. It starts by constructing hierarchical bounding-spheres of the boundaries of the obstacles and the robot. The hierarchy is represented as binary trees. For the obstacles, a tree is constructed to represent all obstacles in the workspace. While for the robot, a tree is constructed to represent each rigid body that constructs the robot. So, a robot with n rigid bodies is represented by n binary trees. Each of these trees keeps a transformation matrix that specifies the position and orientation of the rigid body in the workspace, when the robot is at a particular configuration. The construction of these binary trees is performed before planning starts. During planning, to check whether a configuration q is in-collision, `FreeConf`(q) first places the robot at q by adjusting the transformation matrices attached to each binary tree that represents the robot. Next, Quinlan's method traverses the binary trees as needed to compute the distance between the robot, which may be represented by multiple binary trees, and the obstacles. Whenever two objects collide, Quinlan's method returns zero and `FreeConf` returns false. Notice that this method considers only

the boundaries of the objects. Hence, if an object lies completely inside another object, this method will not detect any collision between the two objects and so does our `FreeConf`. This shortcoming can be overcome by decomposing objects with large volume such that no objects can lie entirely inside another.

A.2 Implementation of `FreePath`

The primitive `FreePath`(q, q') checks whether a straight line segment $\overline{qq'}$ between q and q' is collision-free. For rigid-body robot, we use bisection method [Hsu, 2000]. The idea is to recursively divide $\overline{qq'}$ into two shorter line segments until it is certain that the segment is in-collision or collision-free. We divide the segment until either a segment is found to be in-collision or all segments lie inside collision-free spheres. It starts by using the Quinlan's distance computation to compute the radius r_q and $r_{q'}$ of the largest collision-free spheres centered at q and q'. Next, it divides $\overline{qq'}$ into $\overline{qq''}$ and $\overline{q''q'}$ where $q'' = \frac{q+q'}{2}$ and call `FreeConf`(q'') to check whether q'' is collision-free. If q'' is in-collision, we stop and `FreePath`(q, q') returns false. If $|\overline{qq''}| < r_q$ and $|\overline{q''q'}| < r_{q'}$, where $|\cdot|$ is the length of the line segment, `FreePath`(q, q') returns true. Otherwise `FreePath` recursively divides the two segments and performs the above procedure.

For an articulated robot, the line segment $\overline{qq'}$ is divided recursively until a segment is found to be in-collision or the length of all the segments are less than a small constant ϵ, in which `FreePath`(q, q') returns true.

A.3 Procedure `AllSolved`

Procedure `AllSolved` (Algorithm A.1) finds a path between each pair of the given queries. The queries are represented as a list of 2-tuple $< q_i, q_g >$ of initial q_i and goal q_g configurations. The procedure returns true whenever a path is found for each pair of query. Furthermore, it returns a set of paths in Ψ. Each element Ψ[i] holds the solution of Q[i]. If a query Q[i] can not be solved, then Ψ[i] is NULL. To find the shortest path in $\mathcal{R}$ (Algorithm A.1 line 5), we use Dijkstra algorithm.

Algorithm A.1 `AllSolved`(Q, $\mathcal{R}$, Ψ)

1: N_Q = `sizeOf`(Q).
2: nSolved = 0.
3: Initialize each element of Ψ[i] to NULL.
4: **for** i = 1, ..., N_Q **do**
5: Find the shortest path in $\mathcal{R}$ between Q[i].init and Q[i].goal. If a path is found, increment nSolved by 1 and save the path in Ψ[i].
6: **if** nSolved == N_Q **then** Return true.
7: **else** Return false.