COLLABORATIVE MOTION PLANNING

A Dissertation

by

JORY LONDON DENNY

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

DOCTOR OF PHILOSOPHY

Chair of Committee,	Nancy M. Amato
Committee Members,	Suman Chakravorty
	Dylan Shell
	Dezhen Song
Head of Department,	Dilma da Silva

August 2016

Major Subject: Computer Science

ABSTRACT

Planning motion is an essential component for any autonomous robotic system. An intelligent agent must be able to efficiently plan collision-free paths in order to move through its world. Despite its importance, this problem is PSPACE-Hard which means that even planning motions for simple robots is computationally difficult. State-of-the-art approaches trade completeness (always able to provide a solution if one exists or report none exists) for probabilistic completeness (probabilistically guaranteed to find a solution if one exists but cannot report if none exists) and improved efficiency. These methods use sampling-based techniques to design a sequence of motions for the robot. However, as these methods are random in nature, the probability of their success is directly related to the expansiveness, or openness, of the underlying planning space. In other words, narrow passages, complex systems, and various constraints make planning with these methods difficult. On the other hand, humans can often determine approximate solutions for these difficult solutions quickly.

In this research, we explore user-guided planning in which a human operator works together with a sampling-based motion planner. By having a human-in-the-loop, a human can steer a sampling-based planner towards a solution. This strategy can provide benefits to many applications such as computer-aided design and virtual prototyping, to name a few.

We begin by classifying and creating simple models of common user-guided and heuristic-guided motion planning methods. Our models encompass three forms of user input: configuration-based, path-based, and region-based input. We compare and contrast these approaches and motivate our choice of a region-based collaborative

framework. Through this analysis, we gain insight into user-guided planning and further motivate methods that harness low interface complexity and work entirely in workspace, which is most natural to a human operator. Further, we extend the theory of expansiveness to analyze the various types of user inputs.

Our novel region-based collaboration framework takes advantage of human intuition by allowing a user to define regions in the workspace to bias and/or constrain the search space of a sampling-based motion planner. This approach allows a user to bias a high dimensional search with low dimensional input, supports intermittent user hints, and empowers a user to customize motion solutions.

Finally, we extend region steering to both non-holonomic robotic systems and a human-inspired approach to motion planning.

Our results show that this region-based framework can aid many variants of sampling-based planning, reduce computation time, support solution customization, and can be used to develop advanced heuristic methods for solving motion planning problems. We provide experiments exemplifying our approach in planning motions for complex robotic applications such as mobile manipulators, car-like, and free-flying robots.

DEDICATION

To Katelynn, Pippin, and everyone near and far who supports me through my lifes adventure.

ACKNOWLEDGEMENTS

The research contained in this dissertation and my entire graduate career would not have become a reality without the help of colleagues, friends, and family. You have my sincerest gratitude, and I apologize if I left key players out of the acknowledgements, it was not my intention. This research was also supported in part by a National Science Foundation (NSF) Graduate Research Fellowship (GRFP).

To my advisor, Dr. Nancy M. Amato, I would like to thank you most for your continual advice and support. Since, I started undergraduate research in January 2009, I have come to you for professional advice and attribute many of my successes and awards to you. Specifically, completing my undergraduate thesis, being a finalist for the Computing Research Associations (CRA) Outstanding Undergraduate Research award, earning a NSF GRFP, numerous departmental awards, experiences abroad at conferences, helping me to become an Assistant Professor at the University of Richmond, and now my Ph.D. degree can be traced back to many helpful meetings. I look forward to many future years of advice and collaboration.

To my committee, Dr. Suman Chakravorty, Dr. Dylan Shell, and Dr. Dezhen Song, thank you for conversations at conferences, challenges during my preliminary and final exams, and your support in various endeavors throughout the past few years.

To my future colleagues, Dr. Kostas Bekris, Dr. Dilma da Silva, Dr. Dan Halperin, Dr. Jason O'Kane, and Dr. Lydia Tapia, thank you for professional support and letters of recommendation from which I earned an NSF GRFP and became an Assistant Professor at the University of Richmond.

To my collaborators on collaborative motion planning, fellow graduate student

Read Sandström, undergraduate students Nicole Julian, Brennen Taylor, Andrew Bregger, and Ben Smith, and high school students Jonathan Colbert, Brandon Martinez, Hongsen Qin, and Eli Zamora, thank you for your help on the research contained within this dissertation. Without you, it would have taken much longer, if at all, to complete.

To my collaborators on other projects and friends within the Parasol Lab, I sincerely enjoyed working with you all during my graduate studies and appreciate the fun experiences we shared in lab. I would specifically like to thank Ali-akbar Agha-mohammadi, Chinwe Ekenna, Mukulika Ghosh, Andrew Giese, Adam Fidel, Aditya Mahadevan, Dr. Samuel Rodriguez, Read Sandström, Dr. Timmie Smith, Brennen Taylor, Dr. Shawna Thomas, and Hsin-Yi (Cindy) Yeh.

To my friends inside and outside of the Parasol Lab and to the friendships which did not last, thank you. Each friend, past and present, has taught me something and always shaped my life for the better. I want to specifically name Andrew Giese, Nancy Luong, Zachary Pollock, Brennen Taylor, and Shawn Vicknair, who always manage to stay in contact and support me. I am forever grateful.

To my mom, Barbara, and dad, Gerald, I have not always been the perfect son and have often been distant, but I want to thank you for loving me and supporting me unconditionally. You never said "no" to my endeavors and you never told me that I had to be anyone else than who I was. I love you and I am proud to be your son.

To my brother, Jason, we have had our differences throughout our lives and have never been much for words, but I want to thank you for being a constant source of motivation and let you know that I could not have imagined having a better brother. I love you.

To my dog, Pippin, you will never be able to see these words or understand how

much I love you, but I do. You brought so much joy to my final years of graduate school. You are the perfect dog.

To my wife, Katelynn, I love you more than anything else in the world. You are beautiful and an endless supply of love. You always provide the perfect distraction from the stresses of life, which allows me to remain calm, patient, and resolved. I love all the smiles, laughter, and joy you bring to my life each and every day. I look forward to the next stage of our lives!

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The human mind is a powerful problem solving tool. Many domains leverage this by combining automation with human guidance. Robotics is no exception as both teleoperation [38] and user-guided motion planners, e.g., [45, 8, 59, 94, 102], have been developed to explore these synergies over the past few decades.

Our study focuses on motion planning, or the problem of finding a valid (e.g., collision-free) path for a robot among various robot and/or obstacle constraints. Motion planning has application in robotics, virtual reality [7, 64, 65, 83, 82], bioinformatics [90, 5, 4, 92, 91, 95], and computer-aided design [8, 59, 94, 102], among others.

Over the past few decades, many approaches have been proposed to solve the motion planning problem. Early research classified the general problem in the complexity class PSPACE-Hard [81], which implies there is little hope to find an efficient, complete algorithm for this problem — a *complete* algorithm will find a solution to a problem if one exists or report 'NO' if none exists. In fact, the best known complete algorithm is exponential in the complexity of the robot [13]. Most variants of the motion planning problem are equally difficult. Thus, research has explored other avenues for success, one of which is combining automated algorithms with human guidance.

A specific application that illustrates the potential benefit of combining user-guidance and motion planning is Product Lifecycle Management (PLM), shown in Figure 1.1, which is the process of design, implementation, and maintenance of products. Motion planning can be used in a few portions of PLM. First, in the design phase, virtual prototyping could be used to validate constraints of possible designs.

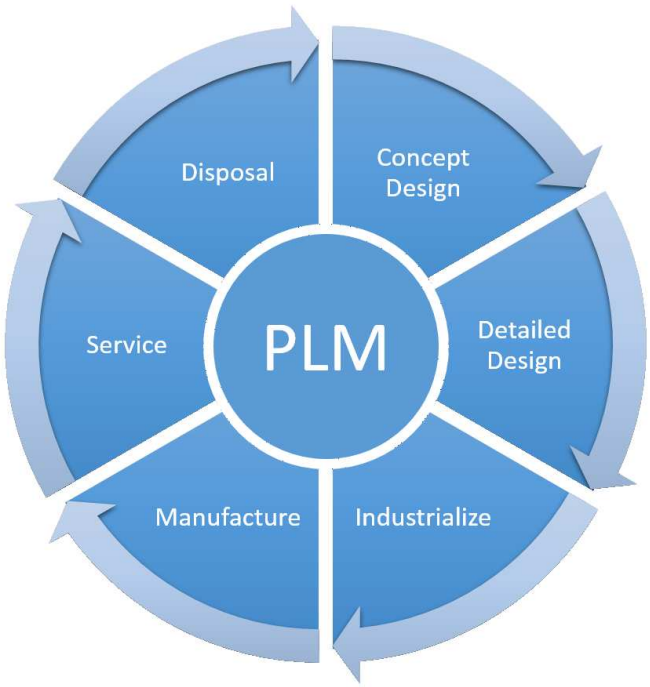


Figure 1.1: Overview of Product Lifecycle Management (PLM). Figure adapted from [102].

Planning is important here to make sure the product can be maintained properly, i.e., parts can be replaced. Second, planning is often found in the manufacturing of products as robots are often used to make products. Third, in the service phase motion planning can be used in determining how to maintain the product. If user input/assistance can be incorporated into PLM, then these phases can be made better and/or more efficient and ultimately cheaper. For example, if user-guided planning is incorporated directly in a computer-aided design tool, then an engineer can validate constraints of the design in a cheap and efficient manner as compared with creating expensive and time consuming physical mock-ups of the product. There are specific challenges to designing a collaborative planner for these applications: (1) collaboration should occur seamlessly in real-time, (2) a user should provide "good" input to

aid the planner — "good" input will aid the planner in identifying the most difficult portions of a planning problem, (3) the planner should plan as close to real-time as possible, and (4) the planner should provide quality feedback on the effectiveness of the user input. This dissertation proposes a novel framework designed to address these challenges and constraints.

There are numerous automated approaches to motion planning. State-of-the-art automated techniques solve this problem through sampling-based methods [53, 43, 61]. These techniques approximate the free planning space by building random graphs, often called roadmaps, that contain representative feasible paths for the robot. As an example, Probabilistic RoadMaps (PRMs) [53], randomly sample the entire planning space and connect nearby samples together to form a roadmap. The map can then be queried for paths through the space by first connecting the start and goal configurations to the roadmap, and then by performing a single-source shortest path search in the graph, e.g., A* [33]. However, it is known that these sampling-based techniques lack efficiency in the presence of narrow passages and difficult robot constraints [42, 52, 40, 43, 58, 41]. In these scenarios, research has aimed at developing intelligent approaches which bias sampling, e.g., [2, 11, 24, 21, 71], yet there are still problems which are difficult to solve efficiently.

User-guided planners encompass methodologies to limit the search space of the planner through user input. By specifying a presumably important (or unimportant) portion of the space, the user enables the planner to focus on a particular subset of the planning problem. Previous user-guided planners address this difficulty by attempting to harness the power of human intuition [45, 8, 59, 94, 102]. In these systems, the human often performs a global scene analysis of the workspace, while the machine handles high-precision tasks such as collision detection and low level path-finding [48, 32]. Recent work has explored sampling-based planning strategies that

incorporate interactive and collaborative planning techniques [94]. These approaches are restricted both to specific sampling-based planners and to an interface which can fully control the robot. To date, no one has come up with theoretically grounded methods for incorporating user input in a planner to solve problems cooperatively, which is the focus of this dissertation.

This dissertation addresses the challenges of providing an effective collaboration mechanism that (1) allows simple user input (not dependent on the complexity of the robot), (2) can work with any sampling-based motion planner, and (3) provides for efficient and effective collaboration and planning. Further, we develop and analyze the theoretical foundations of user-guided planning leading to a deeper understanding of how much and when user-input aids a sampling-based motion planner.

1.1 Research Contribution

In this work, we first seek to classify, model, and compare common user-guided approaches. We classify methods based upon the types of input they gather from the user. In this research, we focus on region-based approaches as they have a good trade-off between interface complexity and performance impact.

We introduce a region-based framework (Figure 1.2), in which a user can collaboratively interact with a sampling-based planner by specifying a workspace region for the planner to prefer or avoid. In turn, this planner can inform the user of its progress and how useful the regions are to the user. Our framework maintains probabilistic completeness of the automated planner used. We show how this framework can be extended for common variants of sampling-based planning.

Our framework is demonstrated in an interactive system that requires only intermittent user action on a standard computer interface, e.g., a mouse. The goal of this work is to understand how these hints and cooperation affect sampling-based

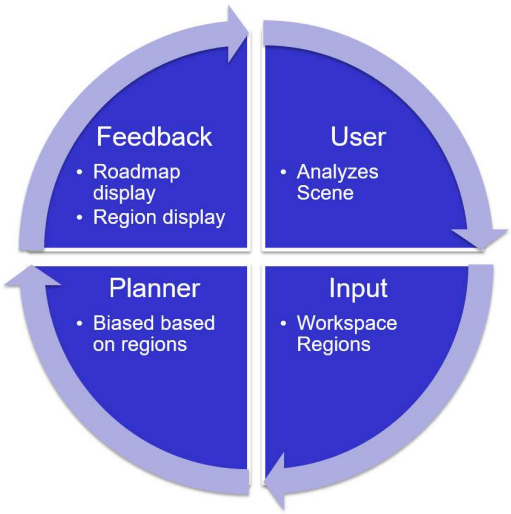


Figure 1.2: Overview of our region-based collaborative framework.

planning. The analysis and optimization of the user interface is the subject of future study.

We analyze each of the user input modalities. To do this, we expand on the concept of *expansiveness* [41] to theoretically analyze the impact these modalities have on the planning process. Our results indicate two things for each user-input modality. First, we define conditions for each user-input method for when the input will aid a sampling-based motion planner. Second, we show that when these conditions are met, the user-input will make a motion planning problem "easier" and lead to more efficient planning for a sampling-based planner.

We show how this region-based framework can be extended to handle complex robot constraints such as non-holonomic constraints, e.g., car-like robots, and how the user-strategies seen in region steering can inspire a novel automated planning approach using geometric techniques for scene analysis.

In summary, this work proposes:

- classification, modeling, and analysis of three types of user-guided input modalities: namely configuration-based, path-based, and region-based input,
- a generic region-based framework for collaborating with any sampling-based planner,
- a theoretical analysis of various user-guided input modalities quantifying both when user input aids in planning and proving that input will make the problem "easier" for a sampling-based motion planner,
- variants of our framework geared toward specific sampling-based planning paradigms, non-holonomic constraints, and a novel human-inspired approach to motion planning,
- and experimental validation of the aforementioned theoretic and algorithmic developments.

Portions of this research were previously published. The basic framework was presented as Region Steering in the proceedings of *the Eleventh Workshop on the Algorithmic Foundations of Robotics* (WAFR) [27]. The variants of our framework applied to graph-based, tree-based, and hybrid sampling-based planning approaches were published in the proceedings of *the Seventeenth International Symposium on Robotics Research* (ISRR) [26]. The discussions of user-guidance input models and theoretical impact of user-guidance on sampling-based motion planning were published in the proceedings of *the International Conference on Intelligent Robots and Systems* (IROS) [25].

1.2 Outline

Chapter 2 discusses important foundational knowledge in representing robots, configuration space, and current approaches to solving the motion planning problem.

Chapter 3 categorizes, models, and compares existing user-guided motion planning algorithms to motivate the algorithmic decisions taken in our region-based user-guidance. Our region-based framework is presented in Chapter 4, and we discuss extensions of the framework to three common paradigms of sampling-based planning: graph-based, tree-based, and hybrid approaches. We revisit the various user input strategies and introduce the concept of augmented (ϵ', α', β')–expansiveness as a means to theoretically study the models in Chapter 5. Then we delve into two interesting extensions of our framework. First in Chapter 6, we extend our framework for handling non-holonomic constraints. Second in Chapter 7, we discuss an automated approach inspired from the user-guidance seen in our approach. Finally, we conclude and discuss future work in Chapter 8.

2. PRELIMINARIES AND RELATED WORK

In this chapter, we discuss motion planning preliminaries and relevant related work.

2.1 Preliminaries

In this section, we discuss preliminary definitions of modeling motion planning problems, and we discuss a few of the basic building blocks for sampling-based motion planning algorithms.

2.1.1 Motion Planning

A robot is a movable object whose position and orientation can be described by n parameters, or *degrees of freedom* (DOFs), each corresponding to an object component (e.g., object positions, object orientations, link angles, and/or link displacements). The following definitions provide the formal specification of this placement, called a *configuration* (Definition 2.1.1), and the set of all placements, called the *configuration space ($\mathcal{C}_{space}$)* [67] (Definition 2.1.2).

Definition 2.1.1. *A **configuration** is a unique placement of the movable object in an environment. It is described by a point $q = \langle x_1, x_2, ..., x_n \rangle$ in an n-dimensional space (where x_i is the i^{th} DOF).*

Definition 2.1.2. *The **configuration space ($\mathcal{C}_{space}$)** is the set of all configurations, feasible or not.*

The $\mathcal{C}_{space}$ may be partitioned into two subsets, *free space* ($\mathcal{C}_{free}$) and *obstacle space* ($\mathcal{C}_{obst}$), Definition 2.1.3 and Definition 2.1.4 respectively. The boundary between the two subsets is referred to as the *contact space* and is denoted by $\partial\mathcal{C}_{obst}$ (Definition 2.1.5).

Definition 2.1.3. *The **free space** ($\mathcal{C}_{free}$) is the subset of all feasible configurations in $\mathcal{C}_{space}$.*

Definition 2.1.4. *The **obstacle space** ($\mathcal{C}_{obst}$) is the union of all infeasible configurations in $\mathcal{C}_{space}$. It can be described as $\mathcal{C}_{space} \setminus \mathcal{C}_{free}$.*

Definition 2.1.5. *The **contact space** ($\partial\mathcal{C}_{obst}$) is the boundary of $\mathcal{C}_{obst}$.*

In general, it is infeasible to compute $\mathcal{C}_{space}$ explicitly [81, 13], but we can often determine the validity of a single configuration quite efficiently, e.g., by performing a collision detection test in the *workspace*, the robot's natural space. Validity is often defined as collision-free (i.e., avoiding both self-collision and collision with the environment) but can be generalized to other constraints such as closure constraints for closed chain systems [60] or energy requirements for computational biology applications [90, 5, 4, 92, 91, 95].

Using the configuration space abstraction, the motion planning problem becomes that of finding a continuous trajectory in $\mathcal{C}_{free}$ from a given start configuration to a goal configuration or region.

2.1.2 Local Transitions

Often in motion planning, we are concerned with proximity between two configurations, which is measured by a *distance metric* (Definition 2.1.6). Many metrics are possible, e.g., Euclidean, Manhattan, or Minkowski distances, and ideally should approximate the cost of transitioning between two configurations. In other words, a lower distance should represent an easier transition between the two configurations.

Definition 2.1.6. *A **distance metric** $\delta(q_1, q_2)$ measures the distance between two configurations q_1 and q_2.*

When two configurations can be connected by a simple path, these configurations are said to be *visible* to each other. Visibility is checked by a *local planner* that returns a sequence of configurations such that all are in $\mathcal{C}_{free}$ and the distance between consecutive configurations is below some resolution threshold or reports that no path is found. Local planners are often deterministic and must either be recomputable or stored. While in principle any local planner can be used, Definition 2.1.7 describes a common strategy, a straight-line interpolation through $\mathcal{C}_{space}$, that is also used in this work.

Definition 2.1.7. *Two configurations q_1 and q_2 are **visible** if $\overline{q_1 q_2} \in \mathcal{C}_{free}$, where $\overline{q_1 q_2}$ is a straight-line interpolation in $\mathcal{C}_{space}$. Let* VISIBLE(q_1, q_2) *be a function which verifies this constraint up to a particular environmental resolution, referred to as a **local planner**.*

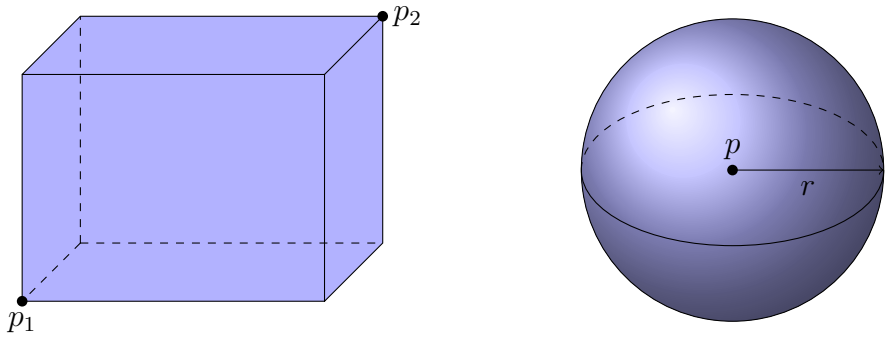
Other common local planners include rotate-at-s [3] (which translates a configuration to a specified percentage, s, of its path towards its goal, rotates it, and then completes the translation to the goal), A* [33], and Toggle Local Planning [23] (which performs local planning in a 2-dimensional triangular subspace of $\mathcal{C}_{space}$).

We define a sequence of local transitions as a *path* (Definition 2.1.8).

Definition 2.1.8. *Two configurations q_i and q_j are adjacent if $\delta(q_i, q_j) \leq r$, where r is an environmental resolution. A **path** $\Pi = \{q_1, q_2, \ldots, q_m\}$ is a continuous sequence of adjacent configurations.*

2.1.3 Regions

We define a *region* as any bounded volume in the workspace (Definition 2.1.9). Two common examples are axis-aligned bounding boxes (AABBs) and bounding spheres (BSs), shown in Figure 2.1.



(a) Axis-Aligned Bounding Box (b) Bounding Sphere

Figure 2.1: Region examples.

Definition 2.1.9. *A **region** R is any bounded volume in the workspace.*

The usage of bounding volumes is not unique to motion planning and can be found in many fields. For example, collision detection libraries use bounding volumes to expedite validity checking of configurations [66].

"Region" can also be seen as a very broad term. A region might be seen as a path or constraint in the workspace to bias planning in $\mathcal{C}_{space}$. For example, one could imagine selecting a wall as a contact-constraint for an end-effector of an articulated robot. There are many possibilities, and we selected a few representative and simple region types for our study.

2.2 Sampling-based Motion Planning

Because of the cost of explicitly computing $\mathcal{C}_{space}$ [81, 13], research has turned to sampling-based techniques to efficiently explore $\mathcal{C}_{free}$ for valid paths. These methods generally fall under two classes: graph-based approaches, e.g., Probabilistic RoadMaps (PRMs) [53], and tree-based approaches, e.g., Rapidly-exploring Random Trees (RRTs) [61] or Expansive Space Trees (ESTs) [43].

Algorithm 1 Probabilistic RoadMap (PRM) Construction

Input: A Environment e
Output: A Roadmap G
1: $G = \{V, E\} \leftarrow \{\emptyset, \emptyset\}$
2: **while** $\neg done$ **do**
3: $\quad V \leftarrow \textsc{Sample}(e)$
4: $\quad \textsc{Connect}(G, V)$
5: **return** G

2.2.1 Graph-based Techniques

Probabilistic RoadMaps (PRMs) [53], shown in Algorithm 1, belong to the class of graph-based approaches. They construct a map of $\mathcal{C}_{free}$ by first randomly sampling valid configurations. Then, nearby samples are connected to form the edges of the roadmap by validating paths using some simple and fast local planner (e.g., a straight-line in $\mathcal{C}_{space}$). This process is repeated until an end condition is reached, e.g., a maximum number of configurations have been sampled. After construction, start and goal configurations are connected to the roadmap (using the local planner) and a graph search, e.g., A* [33], extracts a solution path. PRMs are particularly suited for solving many start and goal queries in the same environment because the roadmap does not need to be recomputed. Various sampling schemes [2, 11, 39, 22, 24, 21] and local planners [46, 3, 24] have been used, and these algorithms are easily implementable, computationally efficient, and applicable to a wide variety of robots.

An important shortcoming of these methods is their poor performance on problems requiring paths that pass through narrow passages in the free space. This is a direct consequence of how the nodes are sampled from $\mathcal{C}_{free}$. For example, using the traditional uniform sampling over $\mathcal{C}_{free}$ [53], any corridor of sufficiently small volume is unlikely to contain any sampled nodes whatsoever [42, 52, 40, 43, 58, 41]. A number of strategies have been proposed to modify the sampling strategy to increase the

number of nodes sampled in narrow corridors.

Obstacle-Based PRM (OBPRM) [2] and Uniform OBPRM (UOBPRM) [21] sample configurations near $\mathcal{C}_{obst}$ surfaces either by pushing configurations to the $\mathcal{C}_{obst}$ boundary or by finding surface intersections of randomly placed line segments. UOBPRM has been theoretically and experimentally shown to generate configurations uniformly distributed on $\mathcal{C}_{obst}$ surfaces.

Gaussian PRM [11] and Bridge Test PRM [39] filter samples with inexpensive tests to find samples near $\mathcal{C}_{obst}$ boundaries or directly in narrow passages, respectively. However, both methods use the same basic sampling as uniform random sampling and suffer from needing many samples to find one in a narrow passage. Additionally, both methods suffer from parameter tuning, which can greatly affect the performance and quality of the mappings produced.

Toggle PRM [22, 24] performs a coordinated mapping of both $\mathcal{C}_{free}$ and $\mathcal{C}_{obst}$. It retains witnesses from failed connection attempts in one space to augment the roadmap in the opposite space. It theoretically and experimentally outperforms uniform random sampling. Toggle PRM can efficiently map "separable" narrow passages, which are those passages that have one less dimension than the overall problem [24].

Some planners transition the problem to other types of spaces to provide more efficient sampling for problems with kinematic constraints, e.g., robots with closure constraints. One method, Reachable Volume sampling [96, 71, 70, 72] performs sampling in Reachable Volume Space, which is an implicit representation of all constraint satisfying samples, in order to plan effectively in these scenarios.

PRM* [51] is a PRM variant that finds asymptotically optimal paths according to some cost function. It differs from PRM in two ways: the number of neighbors considered for connection is a function of the current roadmap size (instead of fixed),

and it attempts all such connections, even if they are already in the same connected component (most PRM implementations ignore such neighbors for efficiency). The cost function is typically path length. PRM* can optimize other cost functions, but these have not been well explored. While PRM* produces asymptotically optimal paths, in practice it requires large roadmaps to do so, and thus is computationally expensive.

Asymptotically near-optimal PRM planners reduce the size of roadmaps produced by PRM* without a significant reduction in path optimality using sparse roadmap spanners [68]. This work filters out edges of the original PRM* graph retaining only those that guarantee that paths between nodes are at most a constant stretch factor greater in cost than the optimal path. While this work reduces the number of edges required, it does not address the issue that all sampled nodes are included in the graph, thus still requiring infinite-sized graphs to meet the near-optimality goals. This infinite node problem is addressed in [28] by only accepting nodes that meet certain coverage, connectivity, and path quality criteria.

2.2.2 Tree-based Techniques

Rapidly-exploring Random Trees (RRTs) [61] and Expansive Space Trees (ESTs) [43] belong to the class of tree-based approaches tailored for solving single-query motion planning problems. RRTs, shown in Algorithm 2, iteratively grow a tree outwards from a root configuration q_{root}. In each expansion attempt, a random configuration q_{rand} is chosen, and the nearest configuration within the tree q_{near} is extended towards q_{rand} up to or at a fixed step size Δq. From this extension, a new configuration q_{new} is computed and added to the tree if and only if there is a valid path from q_{near} to q_{new}. RRTs have been quite useful for a broad range of robotic systems including dynamic environments, kinodynamic robots, and non-holonomic

Algorithm 2 Rapidly-exploring Random Tree (RRT) Construction

Input: An Environment e, a root configuration q_{root}, and a step-size Δq
Output: A Tree T

```
1: T = {V, E} ← {{q_root}, ∅}
2: while ¬done do
3:     q_rand ← GetRandomCfg(e)
4:     q_near ← NearestNeighbor(T, q_rand)
5:     q_new ← Extend(q_near, q_rand, Δq)
6:     T.Update(q_near, q_new)
7: return T
```

robots. They are probabilistically complete and have an exponential convergence to the sampling distribution over $\mathcal{C}_{free}$ because of Voronoi bias, i.e., RRTs explore $\mathcal{C}_{free}$ efficiently. Despite these important properties, RRTs suffer in the presence of narrow passages or very complex planning systems. In this section, we highlight a few of the RRT variants most relevant to this research.

In an effort to solve single query problems faster, RRT-Connect [55] constructs two trees, one rooted at a start configuration q_s and one rooted at a goal configuration q_g. Each tree grows towards the other using a greedy heuristic. Once the two trees meet, a path can be extracted between q_s and q_g using a simple path finding algorithm in the tree. This work also showed the benefit of allowing RRTs to grow with variable step sizes in their greedy heuristic. More specifically, a tree expansion is sometimes allowed to extend until either an obstacle, a maximum expansion distance, or q_{rand} is reached instead of a fixed step size Δq.

There are a few approaches which attempt to limit needless RRT extensions. RRT with collision tendency [17] tracks which inputs have been tried when extending a node to reduce duplicated computations. Once all inputs have been tried, the node is excluded from nearest neighbor selection. Dynamic-Domain RRT [103] biases the random node selection to be within a radius r of q_{near} or expansion will not occur. The

radius is dynamically determined from failed expansion attempts. RRT-Blossom [50] uses the idea of regression constraints to only add edges to a tree which explore new portions of the space. This algorithm also proposes a flood-fill approach when expanding from a node. Reachability-guided RRT [87] looks at reducing needless extensions by only expanding from a node if q_{rand} is closer to the reachability set of a node than the node itself.

Obstacle-based RRT (OBRRT) [84] exploits obstacle information to bias the growth of the tree. Influenced by OBPRM [2] where samples are generated near $\mathcal{C}_{obst}$ surfaces, OBRRT incrementally chooses growth methods based on user-provided probabilities. There are nine growth methods presented including biasing growth with random vectors, random obstacle vectors, tangent obstacle vectors, and even a medial axis biased growth, among others.

Retraction-based motion planning has been explored, primarily to enhance navigation of narrow passages. Retraction-based RRT [104] uses information gained via obstacle contact analysis and optimization to retract RRT growth along the boundary of $\mathcal{C}_{obst}$ to improve RRT performance in narrow passages. It was later extended to handle articulated models [76]. Selective Retraction-based-RRT (SR-RRT) [62] uses tests such as a line bridge-test to focus the expensive retractions to narrow passages and avoid growth in open areas of $\mathcal{C}_{free}$.

Transition-RRT (T-RRT) [49] is a method for growing trees along a cost-map over $\mathcal{C}_{space}$. In this approach, a cost-map is defined over the space, and optimization techniques are used to explore this space. However, T-RRT requires a definition of an allowed transition cost threshold that can be difficult to tune. This method does not guarantee any growth along the medial axis of the space and does not optimize the cost of the path. However, T-RRT has been adopted for obstacle clearance in the workspace and performs well in practice.

RRT* [51] is an approach to ensure asymptotic optimality of the tree. RRT* expands in the same way as RRT except after expansion the tree will locally "rewire" itself to ensure optimization of the cost function. RRT* has been shown to be quite effective in asymptotically finding shortest paths. As with PRM*, it can handle different cost functions, but these have not been explored in the literature. In practice, RRT* requires many iterations to produce near optimal solutions.

2.2.3 Hybrid Techniques

There have been successful approaches combining PRMs and RRTs with the goal of achieving scalability on high performance computers [78] or more effectively exploring narrow passages [86]. Commonly, these approaches use some sort of global sampling over the space, i.e., PRM techniques, to cover $\mathcal{C}_{space}$, and then use RRTs for local exploration to achieve high roadmap connectivity.

RRTLocTrees [93] grows *local trees* rooted at random samples that are unable to easily connect to other trees. RRTLocTrees attempts to connect every new sample to local trees with probability p_{grow}, which tunes the growth of the two global trees (rooted at the start and goal) relative to the local trees. Attempts to merge trees occur when the bounding box of a local tree has grown.

Multi-Modal-PRM [36] can also combine different planning strategies and has shown success in manipulation and legged locomotion applications.

2.2.4 Completeness and Expansiveness

Sampling-based planners are often *probabilistically complete* [19]. The probabilistic completeness property implies that the probability of finding a solution, if one exists, will tend to one as the number of random selections (samples) tends toward infinity. It is a loose guarantee on reliably being able to solve a problem.

Beyond this, some research has explored bounds on the expected number of sam-

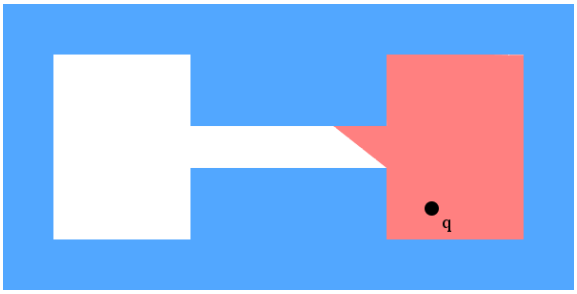


Figure 2.2: Configuration q and its visibility set $V(q)$ (red).

ples required to solve a particular problem instance. One common approach to analysis is through *expansiveness* [41], which we also use in this work. Loosely, if a problem exhibits expansiveness, then uniform sampling will perform adequately. Before we can define expansiveness, we must explore a few preliminary concepts.

The *visibility set* (Definitions 2.2.2 and Figure 2.2) of a configuration or a set of configurations is the subset of $\mathcal{C}_{free}$ for which $\textsc{Visible}(q_1, q_2)$ returns true. Visibility has been a useful concept in sampling-based planners [89] and experimental analyses of sampling-based planners [74].

Definition 2.2.1. *The* ***Visibility Set*** *[41] $V(q)$ of a configuration q is the set of all configurations q' in $\mathcal{C}_{free}$ such that a local planner is able to connect q and q', i.e.,* $\textsc{Visible}(q, q') = true$.

$$V(q) = \{q' \in \mathcal{C}_{free} | \textsc{Visible}(q, q')\}$$

Definition 2.2.2. *The* ***Visibility Set*** *[41] $V(Q)$ of a set of configurations Q is the union of the visibility sets of the individual configurations in Q.*

$$V(Q) = \bigcup_{q \in Q} V(q)$$

.

An essential property of $\mathcal{C}_{free}$ is ϵ–'goodness', Definition 2.2.3, or the ability of all configurations in $\mathcal{C}_{free}$ to be connectible to at least an ϵ proportion of other configurations in $\mathcal{C}_{free}$. From here on, without loss of generality, we assume $\mathcal{C}_{free}$ is one connected component. Let $\mu(X)$ denote the hypervolume, or the measure, of a set X.

Definition 2.2.3. *Let ϵ be a constant in $(0, 1]$. A free space $\mathcal{C}_{free}$ is ϵ–**good** [41] if for all configurations $q \in \mathcal{C}_{free}$ then $\mu(V(q)) \geq \epsilon\mu(\mathcal{C}_{free})$.*

When a $\mathcal{C}_{free}$ is ϵ–good, it is able to be covered by samples in reasonable time. That is, a larger ϵ implies an easier problem because it will be easier to generate samples that cover the planning space.

(ϵ, α, β)–expansiveness is a property of $\mathcal{C}_{free}$ describing how easily the problem can be solved through sampling-based techniques. Being able to sample $\mathcal{C}_{free}$, i.e., ϵ, is only one requirement for sampling-based planners. The other parameters of (ϵ, α, β)–expansiveness, α and β, describe the difficulty of sampling-based planners to generate edges, or motion transitions, through the $\mathcal{C}_{space}$. Put another way, they describe how easy it is to generate configurations that expand the visibility of the roadmap or connect various components of a roadmap representing $\mathcal{C}_{free}$.

First, the β–LOOKOUT, Definition 2.2.4 and Figure 2.3, of a subset X of $\mathcal{C}_{free}$ is the portion of X that can see at least a β fraction of the compliment of X. In this way, β relates to the probability of sampling a new configuration that can expand the visibility of X.

Definition 2.2.4. *Let β be a constant in $(0, 1]$. The β–**lookout**(X) [41] of a set $X \subseteq \mathcal{C}_{free}$ is the set of configurations of X which can connect to at least a β fraction*

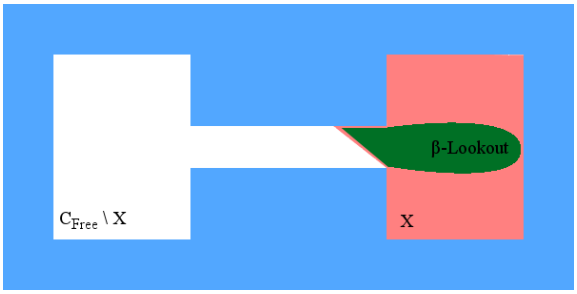


Figure 2.3: Set X (red) and its β–LOOKOUT (green).

of the compliment of X, that is

$$\beta\text{–LOOKOUT}(X) = \{q \in X | \mu(V(q) \setminus X) \geq \beta\mu(\mathcal{C}_{free} \setminus X)\}.$$

With this definition in hand, the full definition of (ϵ, α, β)–expansiveness is given in Definition 2.2.5.

Definition 2.2.5. *Let ϵ, α, β be constants in $(0, 1]$. A configuration space $\mathcal{C}_{free}$ is **(ϵ, α, β)–expansive** [41] if*

1. *$\mathcal{C}_{free}$ is ϵ–good*
2. *For any set M of points, the β–LOOKOUT $(V(M))$ is at least an α fraction of the hypervolume of $V(M)$, that is*

$$\mu(\beta\text{–LOOKOUT}(V(M))) \geq \alpha\mu(V(M)).$$

A higher α and β describe an easier ability to expand and connect a roadmap of $\mathcal{C}_{free}$ through sampling-based planning. (ϵ, α, β)–expansiveness is an inherent property of $\mathcal{C}_{free}$ for a given motion planning problem which cannot be altered. Additionally, these parameters are as difficult to compute explicitly as computing

$\partial\mathcal{C}_{obst}$. Despite this, the parameters can help understand trends in the expected number of samples and how the probability of finding a solution corresponds to the difficulty of the space. Specifically, with probability at least $1-\gamma$, a roadmap of $n = \lceil 16\ln(8/\epsilon\alpha\gamma)/\epsilon\alpha + 6/\beta + 4\rceil$ nodes chosen uniformly at random will be connected, where $\gamma \in (0,1)$ [42].

2.2.5 Workspace-biased Planning Methods

Many planners use workspace information, e.g., regions or decompositions, to aid in the planning process. Here we describe a few relevant frameworks.

Feature Sensitive Motion Planning [73, 75] recursively subdivides the space into "homogeneous" regions (regions of the environment containing similar properties, e.g., free or clutter), individually constructs a roadmap in each region, and merges them together to solve the aggregate motion problem. This framework adaptively determines the specific planning mechanism to map to each homogeneous region, e.g., choosing OBPRM in cluttered regions and uniform sampling in open regions. Similar frameworks have been proposed [85, 97]. RESAMPL samples spherical regions in the space and uses entropy information to decide on the specific planner to use [85]. The Unsupervised Adaptive Strategy (UAS) uses a K-means clustering algorithm to learn important regions of the space and then applies Hybrid PRM in each region [97]. Hybrid PRM uses a reward-based learning mechanism to adaptively select appropriate sampling schemes for a space [44]. A recent approach, Adaptive Neighborhood Connection (ANC), extends Hybrid PRM to connection [30]. ANC could also be incorporated into UAS, but this direction has yet to be explored.

Other approaches utilize workspace decompositions to find narrow or difficult areas of the workspace to bias $\mathcal{C}_{space}$ sampling [98, 57, 79]. These methods begin by decomposing the workspace using an adaptive cell decomposition [98] or a tetrahe-

dralization [57], and then weight the decomposition to bias sampling. However, by automatically identifying regions and disallowing dynamic region specification and modification, the planner might suffer from inefficiencies, such as oversampling in a fully covered region. Additionally, these planners do not typically consider regions that repel sampling.

Some approaches deviate from standard sampling-based methodologies and use heuristic guided search mechanisms built from the A* methodology. One such approach uses an A*-like approach to find a workspace path of overlapping circles, and then uses this to heuristically search for a path composed of predefined kinematic motions [14]. This has been extended to modeling traffic scenarios [15] and dynamic environments [16]. This method however has difficulty extending past 2-dimensional workspaces, car-like robots, and simplistic obstacles.

2.3 User-guided Motion Planning

In many approaches to human-assisted planning, a human operator (user) performs global analysis of the workspace to determine an approximate solution while the machine handles high-precision tasks such as collision detection. In [45, 8, 99] the user can select configurations that are critical to finding a collision-free path, while the planner performs collision checking and path-finding between sub-goals (shown as Cfg Input in Figure 2.4). Certain approaches allow a user to input an approximate path in the scene, and an autonomous planner then morphs this motion into a feasible plan [8, 59, 102] (shown as Path in Figure 2.4). Often, these types of planners have distinct phases for user input and automated planning.

More recently, a two-way communication approach was developed for RRTs, called Interactive-RRT (I-RRT) [94]. In this system, the planner and the user interact in an online fashion to cooperatively solve the problem. I-RRT allows the user

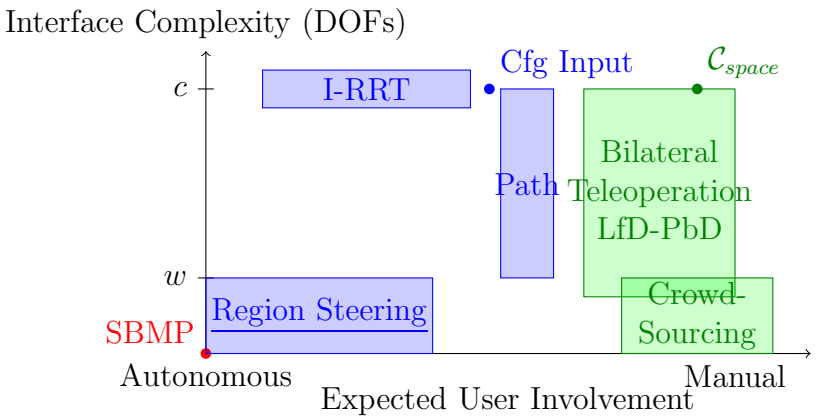


Figure 2.4: Interface complexity in terms of degrees of freedom manipulated versus level of autonomy in expected behavior, where c is the degrees of freedom of the $\mathcal{C}_{space}$ and w is the degrees of freedom of the workspace is shown. The approach described in Chapter 4, called Region Steering here (underlined), is a combination of high autonomy mixed with a simple interface.

to control a robot avatar that biases RRT growth in a virtual scene. This approach, however, is limited to single-query scenarios, requires continuous user input, and is constrained to robotic systems that are fully controllable by the avatar interface (as seen in Figure 2.4).

2.3.1 Crowd-sourcing

There is a distinct but related approach to user-guided planning by which crowd-sourcing is used to steer a large swarm [10] or perform protein folding [9]. In these solutions, a single problem is given to a large number of people in the hopes of finding a global solution or deriving a future control law. However, these approaches have a separate goal compared with user-guided planning, in that user-guided planning allows a user to interact directly with the planner. Despite the different goals however, these approaches can inform each other on best practices in interfacing with a user.

2.3.2 Bilateral Teleoperation

Teleoperation approaches provide closed-loop interactions between an operator and a robot such that the operator has a sense of presence-at-a-distance [38]. Teleoperation focuses on capturing and/or augmenting a user's mechanical skills directly. Often, these approaches try to assist the human operator by predicting where the user is headed, ensuring proper collision avoidance, and maximizing user control over the system [63]. However, these approaches provide minimal automation focusing on presence-at-a-distance allowing human operators to perform tasks in typically dangerous environments, e.g., exploration in space, oceans, or volcanoes. In contrast, human-in-the-loop planning aims to leverage a user's high-level intuition by augmenting an automated planner with information about difficult aspects of the problem.

Nonetheless, both seek to provide a form of two-way communication, referred to as bilateral control in teleoperation literature. Often these approaches have a high interface complexity, e.g., using haptic devices with many degrees of freedom and provide as much control to the user as possible (Figure 2.4). A recent study in teleoperation [69] shows that this form of interaction can be burdensome on the user, e.g., in situations with cyclic or repetitive motions, and takes steps to provide the robot with greater autonomy so that the user need only provide global guidance rather than direct control. Other teleoperation systems allow the user to control a subset of the robot's DOFs through a precomputed $\mathcal{C}_{space}$, as in [48, 47], but these are limited to low dimensional problems. This approach is referred to as $\mathcal{C}_{space}$ in Figure 2.4.

2.3.3 Robot Learning from Demonstration

In Robot Learning from Demonstration (LfD) or Robot Programming by Demonstration (PbD), a human operator trains a robot to perform a new task that the robot has never done before [6]. Typically, this involves a user physically moving the robot through the task. Afterwards, the robot extracts features of this task, i.e., important aspects of the motion or task, and attempts to extrapolate a new control law to complete the task. LfD-PbD research is multi-faceted towards solving various questions such as what tasks should/can be imitated, how to compute/perform an imitation, when can a robot imitate, and whom should the robot imitate [20]? A good example of LfD-PbD is in preparing food using a robot, e.g., pouring ingredients or flipping a pancake [54].

LfD-PbD is in some ways the opposite of teleoperation. In teleoperation, a robot is used to augment a human in various ways, e.g., being physically present on another planet when the human cannot. However, in LfD-PbD a human augments a robots task capabilities by teaching a robot something the human already knows how to do. Naturally, user-guided motion planning is similar to both. In the case of LfD-PbD, a human manually performs the task for the robot upon which a robot then learns and extrapolates information of the task. However, in user-guided planning a higher level of collaboration is sought. There, the human attempts to cooperatively complete a task, e.g., motion planning, splitting the work with the automated abilities of the robot, i.e., the human does high level thinking while the robot does low level computations. Again, we believe that these fields can learn from each other, in the same way user-guided planning might learn from teleoperation and/or crowd-sourcing applications.

3. USER-GUIDED PLANNING

In this chapter, we classify and model common user-guided approaches discussing the interface requirements and trade-offs in the context of sampling-based planning. We augment our discussion with experimental analysis.

3.1 Models of User-guidance

User-guided planners encompass methodologies to limit the search space of the planner through user input. By specifying a presumably important (or unimportant) portion of $\mathcal{C}_{space}$, the user restricts the planner to focus on a particular subset of $\mathcal{C}_{space}$. For example, if a region biases PRM construction, e.g., to a narrow passage, then the planner will focus and build a denser roadmap in the narrow passage as compared with the portions of the $\mathcal{C}_{space}$ not covered by the region. Thus, collaborative planners have the potential to provide more effective planning.

There are a few important design considerations when implementing a collaborative motion planner. First, rendering and feedback should be real-time, or close to real-time, to allow a seamless collaboration. Second, an intuitive mechanism for region specification is needed. Finally, proper user feedback should be customized for the planner. For example, a region might be colored based upon an algorithm-specific perception of usefulness.

Additionally, there is an important trade-off to be made based on PRM time versus the time taken for the user. Essentially, user-guidance can only truly help if the time taken between capturing the user input and planning with it is less than an unaided planner. Our recent work, which is also the core subject of this dissertation (Chapter 4), has shown this is viable for region-based input [27, 26].

From common approaches in the literature, we propose a simple classification

and modeling of user-guided methods into three categories based upon their input modalities: configuration-based input, path-based input, and region-based input.

3.1.1 Configuration-based Input

First, we define a basic model of input where a user places specific configurations in $\mathcal{C}_{free}$ to act as waypoints for a sampling-based motion planner, e.g., [45]. We call these augmenting configurations "beacons." To specify a configuration the interface must have an equal number of degrees of freedom as the robot itself. Hence, it might be prohibitive or expensive to design such an interface which is intuitive and easy to use — we note you could have a user simply specify all DOFs explicitely, but this might not be intuitive.

Because of the precision required and interface complexity related to gathering this information, this methodology of user-guidance may not be ideal in terms of total planning time. However, this input modality will aid in solving a motion planning problem when a user specifies a difficult configuration. In this way, it would provide a divide-and-conquer approach to planning, where the planner solves the problem from the start to each beacon in order and then to the goal.

3.1.2 Path-based Input

In the path-based input model, the user provides a set of paths Π (Definition 2.1.8) in $\mathcal{C}_{free}$. A few methods do allow for path specification approximately [8, 59, 102], and then modify the path to be in $\mathcal{C}_{free}$. However, the input device would require full control of a robot, and might be prohibitive to design a simple and intuitive interface for all robot types.

Here we specifically study a simplified model for our motion planners in which we assume that the end points of each path are added as nodes to the roadmap, and the path is then added as an edge in the same graph. After, a planner progresses as

normal.

3.1.3 Region-based Input

Some approaches allow the user to specify approximate $\mathcal{C}_{space}$ regions, e.g., [94, 27, 26]. The regions are then used to bias the sampling phase of a sampling-based planner to steer planning toward or away from specific portions of $\mathcal{C}_{space}$. An interesting aspect of this type of input is that in order to specify a region of $\mathcal{C}_{space}$ a user does not specifically require an interface with numerous of degrees of freedom. Rather, a low dimensional device can be used (see [27, 26]) — this makes designing interfaces and algorithms with this input modality especially attractive.

Region-based input can be approximate in its specification and still encapsulate knowledge of the difficult portions of the planning space. In problems where the workspace shows strong correlation with the $\mathcal{C}_{space}$ narrow passages, then region-based input is particularly useful. For example, a user can place a region around the workspace narrow passage to effectively guide the planner through the narrow passage.

3.2 Experimental Analysis of User Input Modalities

In this experiment, we study the impact of user-guided planning on PRMs by comparing a PRM's performance with and without the various types of user inputs. Specifically, we explore the effect of the three user models: configuration-based input (Cfg), path-based input (Path), and region-based input (Region). We show that the various user inputs can all aid a sampling-based planner in discovering a solution more efficiently than the planner alone, and we highlight differences in each input's effectiveness. Specifically, we show that, while path-based input might help solve the problem the fastest compared with both configuration-based and region-based input, region-based input allows for a nice trade-off of ease of input and planner efficiency.

3.2.1 Setup

All methods were implemented in a C++ motion planning library developed in the Parasol Lab at Texas A&M University. It uses a distributed graph data structure from the Standard Template Adaptive Parallel Library (STAPL) [12], a C++ library designed for parallel computing.

All experiments were run on a Dell Optiplex 9010 running Fedora 20 with an Intel(R) Core(TM) i7-3770 CPU with 24 GB of RAM with the GNU gcc compiler version 4.8.3.

All methods use uniform random sampling, Euclidean distance, straight-line local planning, and a $k = 10$-closest connection strategy. In all cases of input, high school students provided the key configurations, paths, or regions to satisfy the assumptions of the corresponding models. For configuration-based input, configurations were placed approximately as knowing the exact placements of configurations might not be obvious in a given problem instance. The same input is used for each trial of our experiments. Trials are run until either 10,000 nodes are sampled or a representative query is solved for a scenario.

We compare these different methods in a variety of toy scenarios featuring various types of narrow passages:

- `STunnel` (Figure 3.1(a)) – A simple 2 DOF square robot traverses a difficult S-like narrow passage from top to bottom.

- `ZTunnel` (Figure 3.1(b)) – A 6 DOF cube robot must pass through a difficult Z-shaped narrow passage between the ends of the environment.

- `MazeTunnel` (Figure 3.1(c)) – A 6 DOF spinning-top meanders through a complex series of tubes from top to bottom.

Table 3.1: Success rates of experimental methods.

	STunnel	ZTunnel	MazeTunnel	Walls	Hook
PRM	40%	80%	100%	0%	60%
Cfg	100%	100%	100%	100%	90%
Region	100%	100%	100%	100%	100%
Path	100%	100%	100%	100%	100%

- `Walls` (Figure 3.1(d)) – A 7 DOF articulated linkage must cross a series of narrow passages (walls with holes) from one end of the environment to the other.

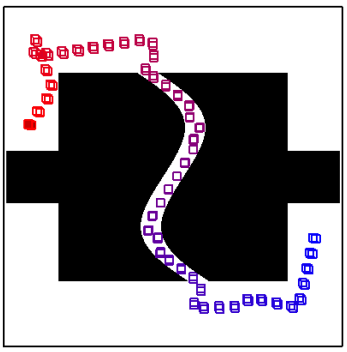
- `Hook` (Figure 3.1(e)) – A 10 DOF free-flying articulated linkage bridges two ends of the environment through a narrow slit in a central wall.

For each trial, we recorded the number of nodes in the final roadmap, the number of collision detection (CD) calls for construction, the time it takes to build, and whether the planner succeeded or not in solving the query in under 10,000 nodes.

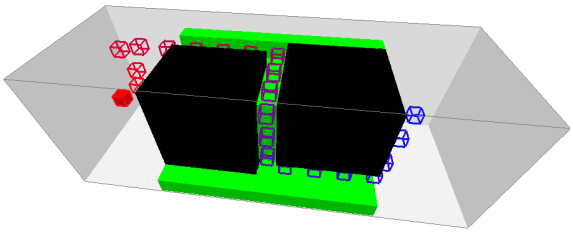
3.2.2 Results

We performed ten trials and reported averages of all runs, successful or not, i.e., we average failed cases (as a best case metric for that specific trial) as well. Success rates are shown in Table 3.1 and metrics are shown in Figure 3.2. We also note that we are not including computations needed for capturing the input, and discuss this trade-off in the discussion below.

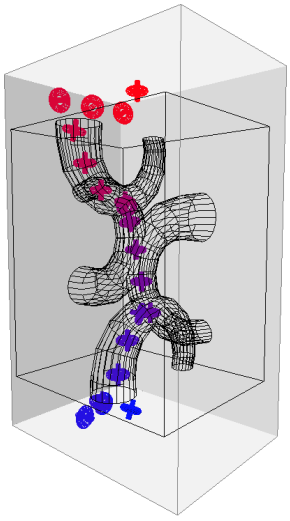
Figure 3.2(a) shows the average number of nodes it took to construct a roadmap to solve the example query. As we can see by this plot, all of the user-based methods solved the scenario with fewer nodes than standard PRM. Figure 3.2(b) shows the average number of CD calls, and Figure 3.2(c) exhibits the average time in seconds each method took to solve the problem. These metrics further indicate that using



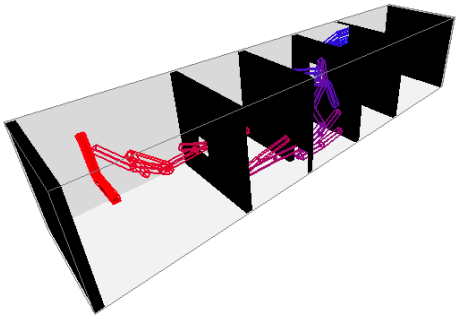
(a) `STunnel` (2 DOF)



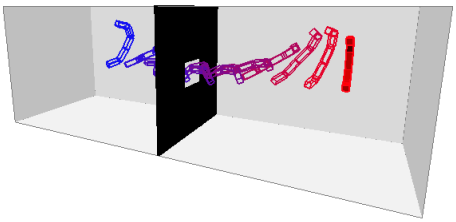
(b) `ZTunnel` (6 DOF)



(c) `MazeTunnel` (6 DOF)

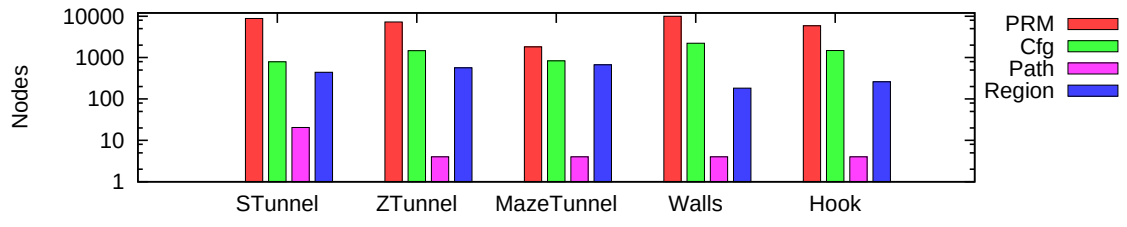


(d) `Walls` (7 DOF)

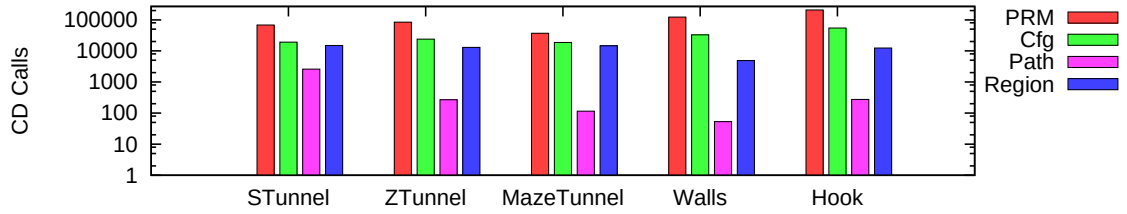


(e) `Hook` (10 DOF)

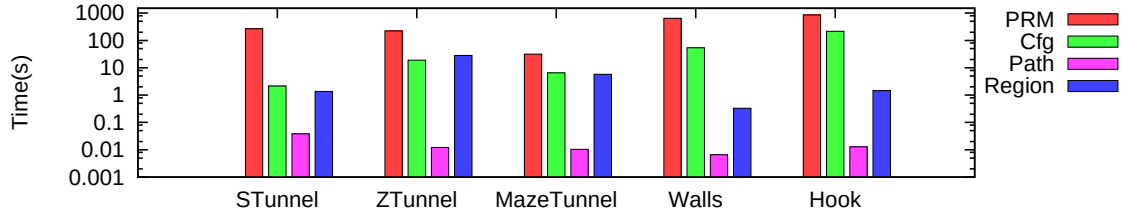
Figure 3.1: Test scenarios. The start (solid red) to end (hollow blue) of an example solution path is shown.



(a) Nodes



(b) CD Calls



(c) Time

Figure 3.2: Experimental results for building a roadmap without user-input (PRM), configuration-based input (Cfg), path-based input (Path), and region-based input (Region). (a) Number of nodes, (b) number of CD Calls, and (c) time in seconds to solve the representative query averaged over ten trials are shown.

user input solves our example problems faster than PRM alone.

An interesting aspect of the data which we discuss more below is that Path generally outperformed Region which in turn outperformed Cfg. The only case this did not hold is in `ZTunnel`. In this case, the specific placements of configurations used avoided the rotational challenge of the narrow passage. They were placed at each turn and reduced the difficulty of the passage to just a translational problem. However, this was not possible with the regions. Moreover, this is a specific narrow

passage and this effect is unlikely to happen in practice.

3.3 Discussion

User-guided planing has been shown to improve over PRM in each of our toy environments. User-guidance helps PRM through the narrow passage by aiding the planner specifically with the most difficult aspects of the problem. That is to say for example with Cfg, the input configurations give PRM waypoints representing the most difficult aspects of the problem. In other words, the area PRM has to search for a solution is reduced by adding critical information from the user. Cfg, Region, and Path all perform well and show that these forms of user input are viable tools to aide future planners.

Figure 3.2 displays an interesting trend that Path required fewer nodes than Region, which requires fewer nodes than Cfg. The main intuition behind this trend is as follows. First, Path input clearly avoids PRM's need to map any configurations in the most difficult portions of a problem. In other words, Path acts as another planner giving a partial solution path to the overall problem reducing the work the PRM has to do. Second, both Cfg and Region still require sampling and connection phases of PRM to finish solving the problem. With Cfg, the need to sample critical configurations is avoided and requires only connection. With Region, a method will still need to sample and connect in difficult portions, but the probability of doing so is altered based on the placement of regions. Finally, Region generally can beat Cfg because it, again, alters the underlying sampling process biasing it towards difficult portions of the space.

4. REGION-BASED COLLABORATIVE PLANNING*

In this Chapter, we describe our general framework for region-based collaborative planning, through which an automated planner and a human operator, referred to as a user, interact to cooperatively discover a solution to a motion planning problem. The general idea is that a user specifies regions of the workspace to bias a planner in $\mathcal{C}_{space}$, and the planner informs the user of its progress by displaying the current roadmap and possibly some indications of region usefulness.

4.1 Framework

Our framework provides a methodology to limit the search space of the planner. By specifying a presumably important (or unimportant) workspace region R, the user restricts the planner to focus on a particular subset of $\mathcal{C}_{space}$. For example, if a region biases PRM construction, e.g., to a narrow passage, then the planner will focus and build a more dense roadmap in the narrow passage as compared with the portions of the $\mathcal{C}_{space}$ not covered by the region. Thus, our collaborative planner has the potential to provide more effective and efficient planning in terms of coverage and plan construction time.

To provide an intuitive mechanism for region specification, we allow a user to specify either Axis-Aligned Bounding Box (AABB) or Bounding Sphere (BS) regions for their simplicity and the ability to specify them with a common mouse interface. Note, we do not claim that the user interface is optimal or intuitive: it is merely sufficient for the user to communicate with the planner and allows us to study the

*The description of the method and some of the experimental results are reprinted from *Algorithmic Foundations of Robotics XI*, "A Region-based Strategy for Collaborative Roadmap Construction," volume 107, 2015, pp. 125–141, J. Denny, R. Sandström, N. Julian, and N. M. Amato,

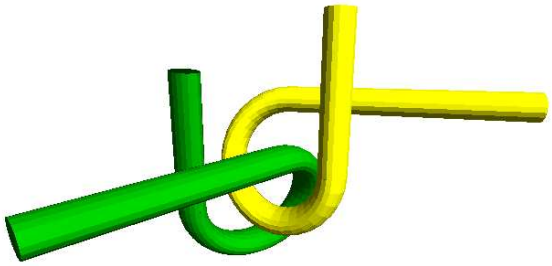


Figure 4.1: Alpha puzzle. A pathologically difficult motion planning problem in which the two α shaped peices must be separated.

usefulness of our collaboration framework. We leave further development of the interface to future work.

While our framework is quite general, it is important to note that our workspace regions are primarily effective in problems where the translational degrees of freedom dominate the system. This occurs often in systems such as mobile robots, unmanned aerial vehicles, or certain CAD applications. In contrast, applying this framework in other scenarios such as strongly rotational problems (e.g., the alpha puzzle, shown in Figure 4.1) would require an alternative region type to successfully limit the planner.

4.1.1 Overview

Our two-way communication framework is shown in Figure 4.2.

In one direction, the user specifies workspace regions to bias the planner. Regions have "types," that can be arbitrarily extended for a particular application. By default, we allow two kinds of regions: attract (bias the planner) and avoid (disallow plans in this region). We allow these regions to be modified at any time, including addition of new regions, resizing/repositioning of current regions, and deletion of regions from the planning scene.

In the other direction, the planner presents its current progress in the scene.

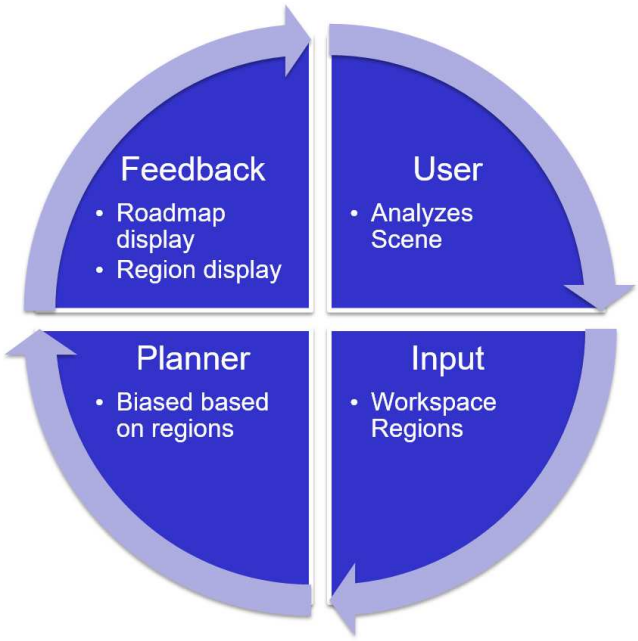


Figure 4.2: In one direction, the user specifies workspace regions, and in the other direction, the planner displays the current progress in planning.

For sampling-based planners this would be the current roadmap being constructed, whether it is a graph or a tree. The tree is drawn by embedding the d-dimensional tree in the 2- or 3- dimensional workspace. Nodes are drawn by a point located at its positional data. Nodes in the same connected component are displayed with the same color so that the user can easily determine whether two nodes are connected. Edges are displayed as polygonal chains of intermittent configurations along the edge. Additionally, we present the current regions to the user. Our framework is able to present a perceived usefulness of regions and recommend regions to the user.

The algorithmic framework for the automated planner shown in Algorithm 3 follows this scheme. Generally, it performs a loop until the planner is finished, e.g., solves an example query or reaches a specific number of nodes. In each iteration, a specific planner biased by the user-defined regions first does automated planning.

Algorithm 3 Region-based Collaborative Motion Planning Framework

Input: Environment e
Output: Roadmap G

```
1: G = (V, E) ← (∅, ∅)
2: while ¬done do
3:    r ← SelectRegion(e.regions)
4:    RegionBiasedPlanner(G, e, r)
5:    G.UpdateMap()
6:    e.UpdateRegions()
7: return G
```

Next, the planner provides feedback to the user through the roadmap and region displays. This is important visual information to help the user adjust regions appropriately.

4.1.1.1 Example

Figure 4.3 shows a simple example that illustrates the general progression of our algorithm in a 2D environment with large obstacles and a few narrow passages. In this example, we create a map-construction query to represent our desired coverage of the space with start and goal configurations. The user begins by specifying particular regions to influence the sampler, as shown in Figure 4.3(a). The user specifies two attract regions (green) in areas that will be difficult for the planner (e.g., a long, narrow passage), as well as one avoid region (striped). The avoid region exemplifies the customizability aspect of our strategy. Though the planner would be likely to sample successfully in that wider passage, the user indicates a desire to avoid that area, perhaps due to environmental considerations not available to the planner.

Over time, the planner identifies one of the attract regions as unproductive and changes its color to red, as shown in Figure 4.3(b). In contrast, the other attract region, within the narrow passage, has proven to be useful and remains green. The

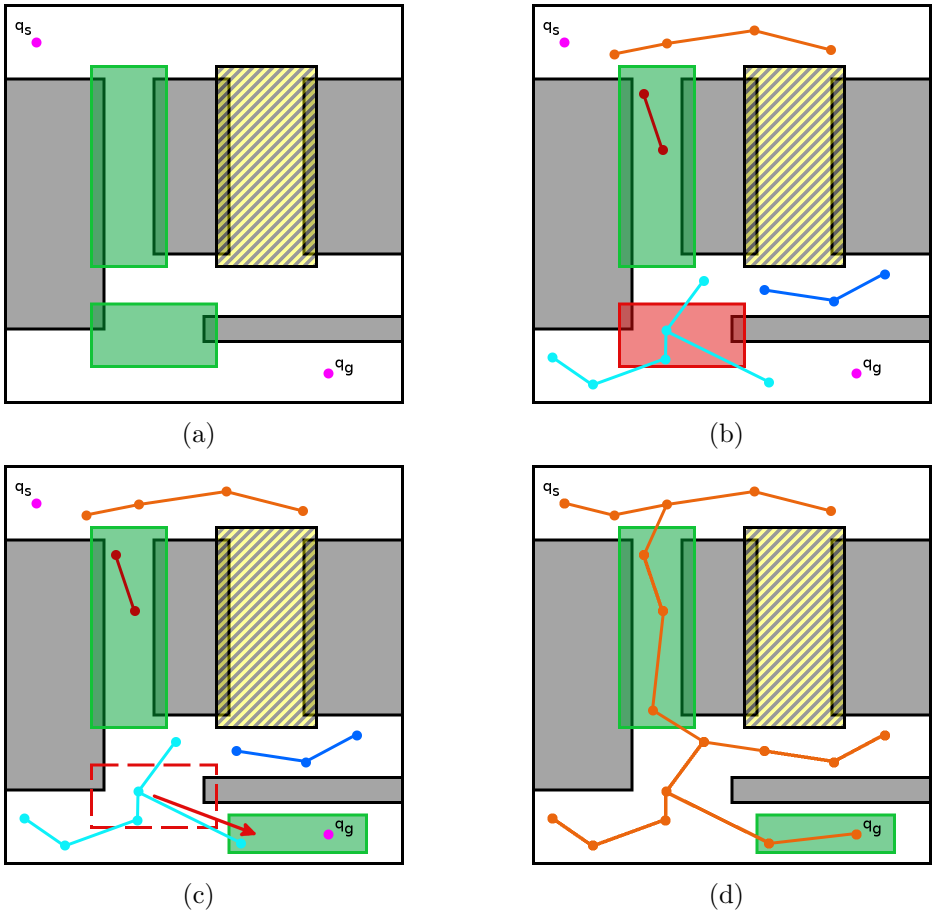


Figure 4.3: Example scenario. (a) A user pre-specifies one avoid and two attract regions. (b) An attract region is shaded red to indicate declining usefulness. (c) The user responds by moving the region to a more productive location. (d) The resulting roadmap.

avoid region behaves as a virtual obstacle and remains devoid of samples. The user appropriately modifies the unproductive region (Figure 4.3(c)) by moving it into the narrow area around the goal and then resizing it accordingly to focus sampling in this area to increase the connectivity of the map.

By exchanging cooperative feedback, the planner and user have discovered the difficult regions of the environment in which to focus node creation. The roadmap can thus be completed and connected efficiently (Figure 4.3(d)).

4.1.1.2 Discussion

We would like to make a note on the overhead of our collaborative framework. Rendering a projected graph in the virtual scene can be expensive for large roadmaps, but this can mitigated by executing the rendering/UI code in a separate thread. Without any user interaction, this results in minimal overhead compared with the un-rendered, unguided planner. Otherwise, the only added cost for our region-based framework is the selection of a region on each iteration, which takes constant time.

Our approach allows the user to customize the roadmap by specifying avoid regions. Avoid regions act like virtual obstacles and block the planner from generating nodes within. By blocking out unwanted workspace areas, the user can easily and intuitively steer the planner toward producing a desirable roadmap. For example, suppose our system is used to plan motions for a robot surveying an area. The user can alter the roadmap in real time by specifying dangerous areas as avoid regions that the robot must evade. This flexibility offers an efficient means for handling transient or previously unknown hazards as the roadmap can be modified without needing to conduct further sampling.

4.1.2 User Input

In our collaborative system, we allow various forms of input to manipulate the regions in an online and interactive fashion. First, the user can pre-provide regions to the planner before planning begins. Second, the user can add, delete, move, and resize regions during the planning process. Finally, the user can optionally handle the regions which are recommended by the system. All of these options are constructed to avoid the need for continuous interaction: the user can provide as much or as little input as desired.

We use simple mouse input to accommodate the various forms of interaction.

Based upon where the user clicks, we can project the 2D window coordinate $w = \langle w_x, w_y \rangle$ to a 3D plane defined by a point $p = \langle p_x, p_y, p_z \rangle$ and a normal $\vec{n} = \langle n_x, n_y, n_z \rangle$. We use this operation to allow intuitive region definition and manipulation. We outline all of the operations on regions below:

4.1.2.1 Addition

When adding a region, the user can click in the scene to define a vertex of the bounding volume and drag the mouse to size appropriately. In planar environments, the mouse position is projected directly onto the environment plane. For volumetric environments, the mouse position is projected onto the plane defined by a point $p = cam_{pos} + d * \overrightarrow{cam_{dir}}$ and a direction $\vec{n} = -\overrightarrow{cam_{dir}}$, where cam_{pos} is the position of the camera, $\overrightarrow{cam_{dir}}$ is the direction the camera is facing in the scene, and d is a displacement distance (typically $^1/_3$ of the environment's bounding radius). For example, to add an AABB region, the user clicks the scene, which defines a single vertex, and then drags the mouse to size the box and define a second, opposite vertex to complete the AABB (shown in Figure 2.1).

4.1.2.2 Deletion

The user can select any region at any given point in time. If the user selects a region, it can be ordered for deletion. Selection is based upon projecting the mouse position into the scene to identify the object it hits first.

4.1.2.3 Manipulation

Manipulation is a bit more difficult. All regions can be translated and resized in the scene. When translating, we allow for two motions. If the user left-clicks the selected region, we translate on the plane defined by $p = c$ and $\vec{n} = -\overrightarrow{cam_{dir}}$, where c is the center of the region. If the user right-clicks the selected region, then we

translate in and out along $\overrightarrow{cam_{dir}}$. To resize a given region, the user highlights the edge of the region and resizes via click-and-drag. For example, with AABB regions, selecting an edge allows manipulation for two of three dimensions at any given time, or for BS regions, the radius can be manipulated by selecting the boundary of the projected sphere. When a region is manipulated, the numbers of successful and failed sampling attempts are reset so that the region's effectiveness and coloring can be recomputed.

4.1.2.4 Recommendation Processing

When the user sees a recommended region, which is initially proposed, the user can ignore the region completely, delete it, or manipulate and commit it as either an attract or an avoid region. Thus, these regions do not affect the planner until they are handled by the user.

4.1.3 Completeness

To retain the completeness properties of the underlying planner, we consider the entire workspace as a region. Drawing a sample from this region is identical to drawing a sample from the entire $\mathcal{C}_{space}$, which is equivalent to the behavior of the underlying unguided planner. By having a probability to select this global region, we inherit the probabilistic completeness property of the underlying planner(s) — our framework can be extended to handle multiple planners easily, for example by planning with both in parallel.

There are two cases in which our framework cannot guarantee probabilistic completeness. The first is when the underlying planner is not probabilistically complete — while the user may still be able to manipulate regions to solve the problem, a solution is not guaranteed. The second is when the user places an avoid region that changes the topology of $\mathcal{C}_{free}$. Recall that avoid regions are hard constraints, i.e.,

Algorithm 4 Region-biased PRM

Input: A Roadmap G, an Environment e, and a Region r

```
1: q ← r.sampler.SAMPLE(r)
2: if q ∉ a, ∀a ∈ e.avoidRegions then
3:    G.ADDANDCONNECT(q)
4:    if ISDIFFICULTNODE(q) then
5:       e.RECOMMENDREGION(q)
```

placing such a region is equivalent to placing a virtual obstacle in the scene. Hence, the user can make the problem unsolvable by placing avoid regions to block all available solutions. This is a powerful tool that requires some discretion. While the user can unintentionally block valid solutions, they can also employ avoid regions to block out unimportant areas of the workspace. In the future, one could expand our framework to allow avoid regions to be soft constraints or modify the planner to find the minimally invasive plan [35].

4.2 Framework Variants

In this section, we show three variants of our framework: collaborative region-biased roadmap construction, collaborative region-biased tree construction, and a collaborative region-biased hybrid method.

4.2.1 Collaborative Region-biased Roadmap Construction

This variant, called Region-biased PRM, is a collaborative roadmap construction approach in which a user specifies regions to bias a graph-based planning method, e.g., PRM. Shown in Algorithm 4 and Figure 4.4, Region-biased PRM allows the user to select a different sampler for each region. For example, if the user specifies a region around a narrow passage, they could also direct the planner to use obstacle-based sampling [2] within that region.

During sampling, an attract region is first selected at random (recall that the en-

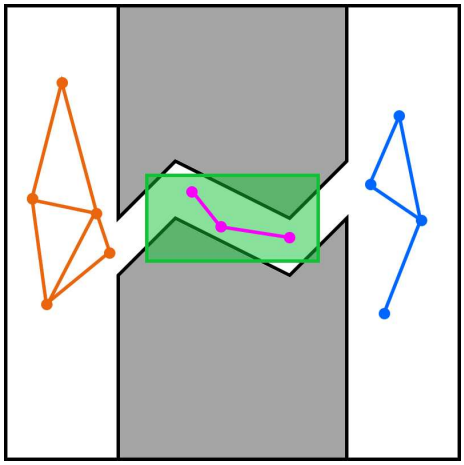


Figure 4.4: A user has drawn a region to mark the narrow passage and set the sampler to OBPRM [2] in the region. Region-biased PRM uses this to bias roadmap construction to the narrow passage.

tire environment is also considered an attract region). Then, a sample is generated within that region by selecting a random point within the region to set positional DOFs and randomizing all other DOFs. Afterwards, a user-specified sampling mechanism, e.g., OBPRM [2], is applied to the region-biased sample. We additionally require that when the robot is placed at the random configuration that all points of the robot are still located within a region. The reasoning is to provide a predictable behavior for the user. Once a sample is created, it is checked to ensure it does not lie within any avoid regions. If the sample meets the criterion, it is added and connected to the roadmap. If the sample fails to connect, i.e., it is in a difficult area of $\mathcal{C}_{space}$, the planner can additionally recommend regions to the user. After each iteration, the perceived usefulness of each region to the planner is shown.

To guide the user's manipulation of the regions, we color the regions based upon their perceived usefulness u to the planner by setting the region's RGB value to be $\langle 1-u, u, 0\rangle$. In this manner, the region is green when it is most useful and red when

it is least productive. We base the usefulness on the approximated density d of the successful samples within the region in $\mathcal{C}_{free}$:

$$d = \frac{n}{\mu(\mathcal{C}_{free} \cap r)} \approx \frac{n}{\mu(r)\frac{n}{n+f}} = \frac{n+f}{\mu(r)},$$

where n is the number of successful samples, f is the number of failed sampling attempts, and $\mu(r)$ is the volume of the region r. Essentially, we are loosely approximating the ratio of successful samples to the volume of $\mathcal{C}_{free}$ covered by r. We define usefulness as $u = e^{-d^2}$, which allows a smooth transition from useful to unproductive region coloring. Our choice in metric is a monotonically decreasing function over time motivated by the fact that too many samples in $\mathcal{C}_{obst}$ do not add anything to the roadmap and too many samples in $\mathcal{C}_{free}$ create oversampling and again do not greatly help the planning process. It is important to note, other metrics could be used to indicate usefulness and achieve various goals. For example, if the goal is to provide samples of high clearance (i.e., large distances from obstacles), then usefulness might be defined as the clearance of the center of the region. We chose our metric to specifically indicate success of sampling to aid a user in discovering the solution faster.

4.2.2 Collaborative Region-biased Tree Construction

Our region-based framework can also be extended to bias tree-based approaches, e.g., RRT. Our algorithm, called Region-biased RRT, is shown in Algorithm 5 and Figure 4.5. The algorithm proceeds as a typical RRT by selecting a configuration q_{rand}, though in this case the selection is biased by a region. First, the planner selects a random region and generates a new configuration q_{rand} within that region (using the same process as with Region-biased PRM). Then, the nearest node in the tree q_{near} is determined, and a node q_{new} is generated by extending q_{near} towards q_{rand}.

Algorithm 5 Region-biased RRT

Input: A Roadmap G, an Environment e, and a Region r

1: $q_{rand} \leftarrow r.\text{GETRANDOMCFG}()$
2: $q_{near} \leftarrow \text{NEARESTNEIGHBOR}(G, q_{rand})$
3: $q_{new} \leftarrow \text{EXTEND}(q_{near}, q_{rand}, \Delta q)$
4: $G.\text{UPDATE}(q_{near}, q_{new})$

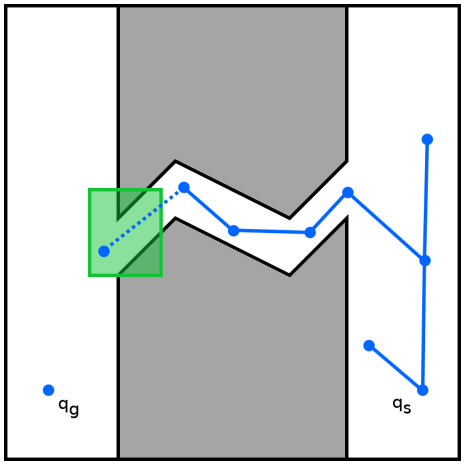


Figure 4.5: A user has drawn a region to act as a waypoint for the RRT, influencing Region-biased RRT to grow through the narrow passage.

In our system, this extension is not permitted to extend through avoid regions in the environment.

4.2.3 Collaborative Region-biased Hybrid Methods

Here, we extend a recent hybrid approach called Spark PRM [86], which is essentially a PRM planner that grows or "sparks" RRTs in narrow passages to increase roadmap connectivity. Using our framework, a user can specify regions to control where RRTs are sparked in the environment to aid PRM connection. These sparked RRTs are grown until they connect to the roadmap or a maximum number of iterations is reached. Our algorithm, Region-biased Spark PRM, is shown in Algorithm 6

Algorithm 6 Region-biased Spark PRM

Input: A Roadmap G, an Environment e, and a Region r

1: $q \leftarrow r.sampler.\textsc{Sample}(r)$
2: **if** $q \notin a, \forall a \in e.avoidRegions$ **then**
3: $\quad G.\textsc{AddAndConnect}(q)$
4: $\quad$ **if** $r \neq e \wedge \textsc{InNarrowPassage}(q)$ **then**
5: $\quad\quad G \leftarrow G \cup \textsc{ConstructRRT}(q)$

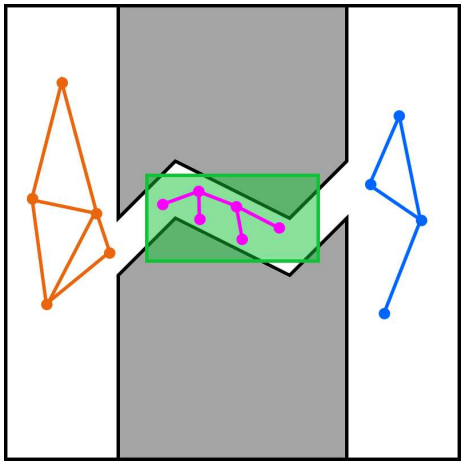


Figure 4.6: A user has created a region to mark a narrow passage and Region-biased Spark PRM begins growing a RRT within.

and Figure 4.6.

4.3 Experimental Analysis of Variants

In this section, we evaluate the collaborative planners in two scenarios, involving a 2 DOF omnidirectional robot and an 8 DOF mobile manipulator measuring speedup compared to their fully automated counterparts.

4.3.1 Setup

All methods were implemented in a C++ motion planning library developed in the Parasol Lab at Texas A&M University. It uses a distributed graph data structure

from the Standard Template Adaptive Parallel Library (STAPL) [12], a C++ library designed for parallel computing.

All experiments were run on a Dell Optiplex 9010 running Fedora 20 with an Intel(R) Core(TM) i7-3770 CPU with 24 GB of RAM with the GNU gcc compiler version 4.8.3.

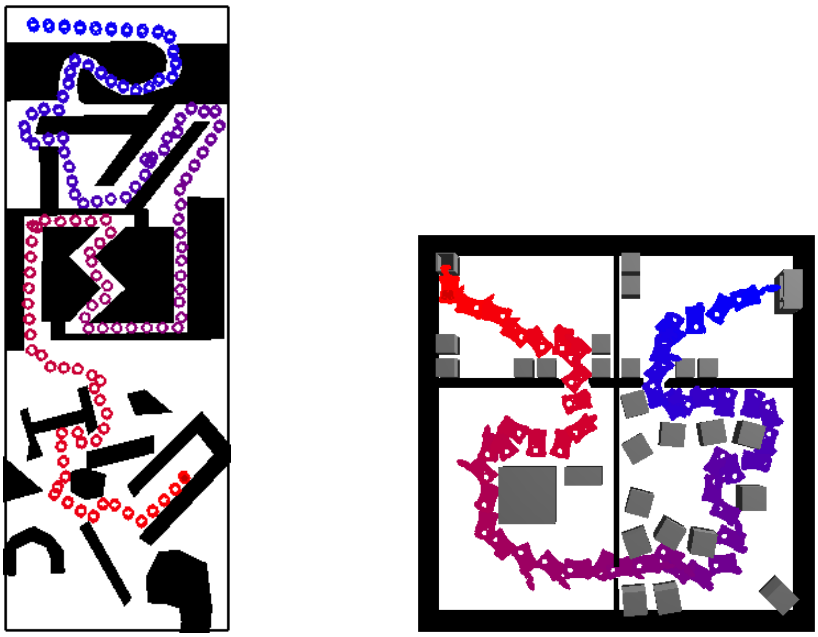
We evaluate each method in two scenarios as seen in Figure 4.7. Queries are shown in start configuration (red) and goal configuration (blue) pairs.

- In `Planar` (Figure 4.7(a)), a planar 2 DOF robot must traverse a series of difficult narrow passages and cluttered areas from the left to the right of the environment.

- In `Manipulator` (Figure 4.7(b)), a simulated 8 DOF KUKA youBot [56] begins by reaching into an open box. It must then pass through doorways while navigating around boxes in an industrial scene. The query ends with the robot reaching into a cabinet on a table. This robot has an omnidirectional base and an arm with five joints.

We are interested in the success rate and the total time for the planner to solve each scenario (including user input and collaboration time). Experiments are run with 10 trials, and the metrics reported are averages of the successful runs.

The user-guided executions were performed by graduate and undergraduate students studying motion planning. In order to minimize the impact of user variance, the same user performed all of the executions in a given environment. Additionally, the users were allowed to practice with the system until they developed familiarity with the interface and environments. Recall, as mentioned previously, this research is focused on the usefulness of user collected information and not on the particular interface.



(a) `Planar` (2 DOF) (b) `Manipulator` (8 DOF)

Figure 4.7: Example scenarios used in experimental analysis. (a) A planar 2 DOF scenario. (b) An 8 DOF mobile manipulator. All queries require traversal through narrow passages between the start (solid red) and goal (hollow blue) configurations.

4.3.2 Region-biased PRM

In this experiment, we compare Region-biased PRM with other common PRM sampling techniques in order to show its ability to efficiently compute a roadmap.

4.3.2.1 Setup and Results

In this experiment, we analyze Region-biased PRM by comparing its performance with Basic PRM (referred to as Uniform) [53], OBPRM [2], and Gaussian PRM [11] (referred to as Gaussian). We analyze Region-biased PRM with a homogeneous use of uniform random sampling (referred to as RBPRM-U) and a heterogeneous use of uniform and obstacle-based sampling (RBPRM-H). For Gaussian PRM, we use both a tuned and untuned d value of the Gaussian distribution based upon twice

Table 4.1: Success rates of various PRM construction methods.

Planner	`Planar`	`Manipulator`
Uniform	0%	100%
Gaussian-T	90%	100%
Gaussian-U	40%	100%
OBPRM	10%	100%
RBPRM-U	100%	100%
RBPRM-H	100%	100%

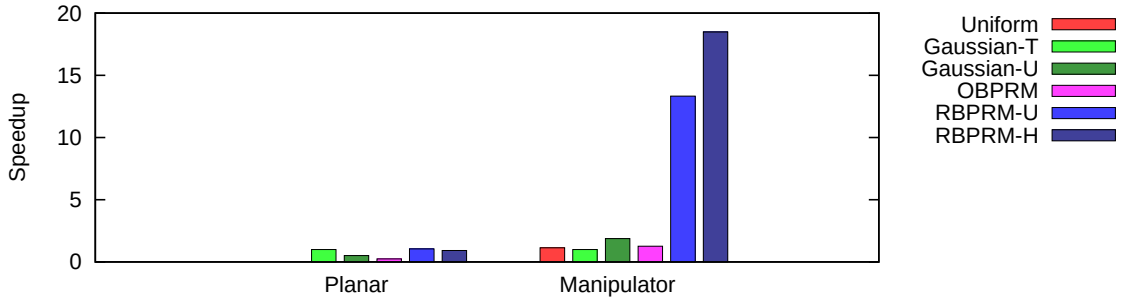


Figure 4.8: Speedups of various PRM construction methods compared with a tuned Gaussian PRM (Gaussian-T).

the robot radius for the environment (referred to as Gaussian-T and Gaussian-U respectively). All methods use Euclidean distance, straight-line local planning, and a $k = 10$-closest neighbor connection strategy.

Each planner is run until either a construction query is solved or 5,000 nodes are sampled. The construction query is designed to verify that the roadmap well-represents $\mathcal{C}_{free}$ by requiring connectivity through the major areas of the environment. Success rates are shown in Table 4.1 and speedups compared with Gaussian-T are shown in Figure 4.8.

4.3.2.2 Performance Comparison

In terms of success rates, Region-biased PRM outperforms each method solving both problems 100% of the time. Even including the interaction time, Region-biased PRM generally solves each problem faster when compared to the fully automated approaches. In the easier environment, using a tuned Gaussian PRM has comparable performance to ours. Typically, we saw improvements up to two times compared with the best PRM performance in each problem. We note that tuning Gaussian PRM can be difficult and require running the problem many times to find an optimal value. Region-biased PRM is capable of acquiring more consistent results because the user can visually determine where the PRM has yet to map in the environment and focus planning there.

4.3.2.3 User Strategy

In `Planar`, the user's strategy generally involved allowing the planner to progress for a second, identifying the narrow passage where configurations were scarce, and using regions to patch the roadmap in these areas. Similarly, the user's strategy in `Manipulator` was to create attract regions wherever the planner had not yet connected in the first three quadrants of the environment (i.e., those quadrants not containing the goal). However, the strategy changed for the goal quadrant because it contained a joint-posturing aspect. Here, the user employed avoid regions to remove unproductive configurations that sampled near the goal but did not connect to it. This improved the likeliness that a connectible configuration would see the goal as a nearest neighbor, thereby expediting the difficult task of connecting the goal to the map. The user deleted or moved attract regions to new locations as soon as the roadmap connected through them to limit redundant configurations.

Table 4.2: Success rates of various RRT construction methods.

Planner	`Planar`	`Manipulator`
RRT	10%	90%
OBRRT	90%	100%
I-RRT	100%	–
RBRRT	100%	100%

4.3.3 Region-biased RRT

In this experiment, we compare Region-biased RRT with other common RRT growth methods in their effectiveness in solving queries.

4.3.3.1 Setup and Results

We compare Region-biased RRT (referred to as RBRRT) with RRT [61], OBRRT [84], and I-RRT [94]. We use $\Delta q = 10$ which is approximately 10% of the diagonal of each environment. With OBRRT, we use appropriate distribution of growth methods for each environment, but note there could be a more effective parameterization to optimize performance. I-RRT's parameters were selected based on recommendations in [94]. We only compare against I-RRT in the `Planar` environment because this is the only robot fully controllable by our interface, a mouse with 2 DOF (recall that I-RRT requires a 1-to-1 mapping between interface and robot DOF). Success rates are shown in Table 4.2 and speedups compared with RRT are shown in Figure 4.9.

4.3.3.2 Performance Comparison

The interactive methods were able to solve both scenarios 100% of the time, whereas the automated planners were not able to in `Planar`. Unguided RRTs performed poorly because the maze-like shape of the problem's narrow passages made growth difficult. In contrast, the user directed the expansion of the tree through the passages in the interactive approaches. Compared with RRT, both I-RRT and

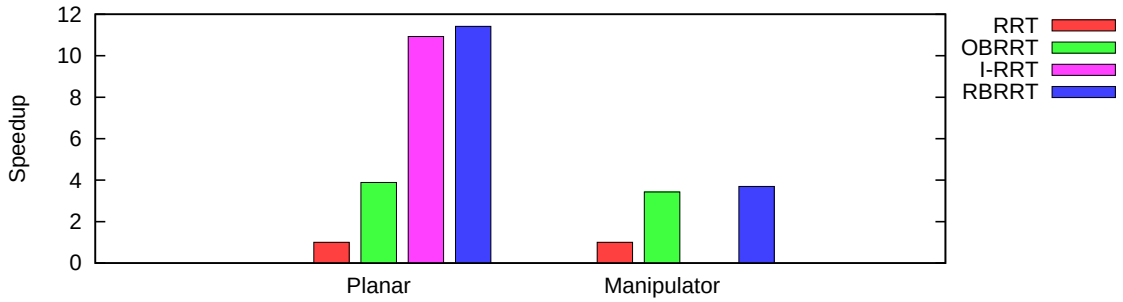


Figure 4.9: Speedup of various RRT techniques compared with RRT.

RBRRT had speedups of up to 12 times in `Planar`, and RBRRT had speedups of up to four times in `Manipulator`. These results show the power of the interactive methods when compared to the fully automated approaches.

4.3.3.3 User Strategy

The user's strategy in both problems was to start the planner immediately and drag a single attract region through the environment, staying just ahead of the tree growth. This differed from the PRM-based strategies because RRTs always grow outward from the existing roadmap. This leading strategy allowed the user to effectively guide the RRT through narrow passages. The attract region used was roughly twice the size of the robot's base, which provided a good mix of flexibility and precision for steering the RRT. This is similar to the strategy used by I-RRT which explains their comparable performance in `Planar`.

4.3.4 Region-biased Spark PRM

In this experiment, we compare Region-biased Spark PRM with another hybrid technique to again show the extensibility of our framework and exemplify how it can make planning more efficient.

Table 4.3: Success rates for Region-biased Spark PRM and Spark PRM.

Planner	`Planar`	`Manipulator`
Spark PRM	90%	100%
Region-biased Spark PRM	100%	100%

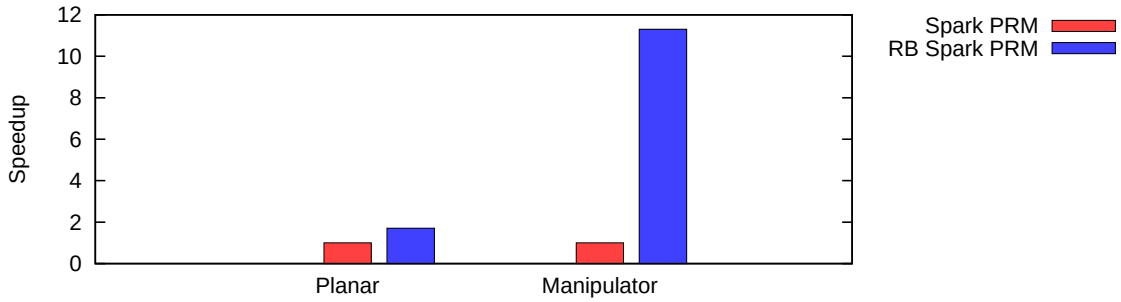


Figure 4.10: Speedup comparison between Region-biased Spark PRM and Spark PRM.

4.3.4.1 Setup and Results

Here, we compare Region-biased Spark PRM (referred to as RB Spark PRM) to Spark PRM [86]. Parameters for Region-biased Spark PRM and Spark PRM are identical and based upon recommendations in [86]. Both methods use Euclidean distance, straight-line local planning, and a $k = 10$-closest neighbor connection strategy.

Each planner is run until either a construction query is solved (identical to the Region-biased PRM experiments) or 5,000 nodes are sampled. Success rates are shown in Table 4.3 and speedups compared with Spark PRM are shown in Figure 4.10.

4.3.4.2 Performance Comparison

In this experiment, we can see that RB Spark PRM was able to focus Spark PRM's planning process effectively to provide more consistent and faster results, up

to nine times speedup. The interactivity of these methods and our framework is extensible enough to speedup very efficient hybrid approaches to planning.

4.3.4.3 User Strategy

The user's strategy for RB Spark PRM was similar to that of Region-biased PRM. The primary difference in `Manipulator` was that fewer attract regions were required as the sparked RRTs quickly bridged unconnected components of the roadmap.

4.4 Experimental Analysis of Collaboration Loop

The previous section established that our framework is widely applicable to many sampling-based planning approaches. In this section, we look in-depth at the collaboration loop and region shape as it influences one of the variants of our framework, Region-biased PRM.

4.4.1 Setup

In our experiments, we study the impact of Region-biased PRM on PRM sampling techniques by comparing its performance with Basic PRM (referred to as Uniform) [53], OBPRM[2], and Gaussian PRM[11] (referred to as Gaussian), and I-RRT [94]. Our strategy is not restricted to any underlying sampling technique. We use uniform random sampling in these experiments, but any sampler can be used. As such, we believe it is fair to compare against other samplers which bias sampling for narrow and cluttered environments, such as OBPRM and Gaussian PRM. For Gaussian PRM, we configure the d value of the Gaussian distribution to twice the robot radius for the environment, which provided consistent results. Though there may be better d values, we believe that this maintains a fair basis of comparison. I-RRT's parameters were selected based on recommendations in [94]. We test Region-biased PRM with both AABB regions, referred to RB-AABB, and BS regions, referred to

as RB-BS. Additionally, we test Region-biased PRM using both one-way and two-way interaction to demonstrate the benefit of two-way collaboration. In the one-way tests, all regions are input prior to the PRM execution, i.e., the user is not allowed to alter any regions during mapping. In the two-way tests, the user is allowed to add, alter, and delete regions during roadmap construction as they see fit. All methods use Euclidean distance, straight-line local planning, and a $k = 10$-closest neighbor connection strategy.

The user-guided executions were performed by graduate students studying motion planning. To minimize the impact of user variance, the users were allowed to practice with the system until they developed consistent performance. Consequently, one- and two-way strategies did not vary significantly across trials and in many cases differed primarily in greater care taken in region creation for one-way tests and ability to delete unproductive regions in two-way tests. However, it should be noted that in practicing with the system the users were able to receive feedback from the planner on their one-way strategies; no such feedback would be available in a true one-way system. The one-way tests thus represent the idealized performance of a user who knows an effective strategy *a priori*. Recall, as mentioned previously, this research is focused on the usefulness of user collected information and not on the particular interface.

For these experiments, roadmap construction halts after solving a construction query or sampling ten thousand nodes. The construction query is designed to verify complete coverage of the environment such that if the query can be solved using the roadmap, then the roadmap sufficiently covers $\mathcal{C}_{free}$. Failing to solve the query indicates that there are areas that are disconnected or not covered in the roadmap. Thus, the roadmaps constructed by Region-biased PRM are reusable for multi-query use, i.e., after the initial query, subsequent queries can be solved with minimal or no

further sampling. We report the number of successful completions, the number of nodes in the final roadmap produced, the time required for initial user input (for our collaborative region strategy), and the time needed to build the map. All experiments are run with 10 trials, and the metrics reported are averages of the successful runs.

Environments are shown in Figure 4.11. Construction queries are shown in start configuration (red) and goal configuration (blue) pairs.

- In `Heterogeneous` (Figure 4.11(a)), a simple 2 DOF robot must traverse a series of cluttered regions and narrow passages from the top to the bottom of the environment.

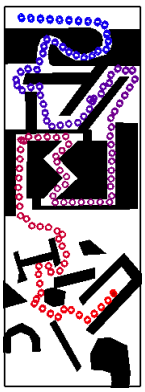
- In `FloorPlan` (Figure 4.11(b)), a 3 DOF mobile robot must traverse through a cluttered apartment from a living room to a bedroom. This environment is representative of a possible robotic assisted-living platform for retirement communities, upon which the floor plan is based.

- In `LTunnel` [1] (Figure 4.11(c)), an L-shaped, 6 DOF free-flying robot must traverse two difficult narrow passages to get from the left side of the environment to the right.

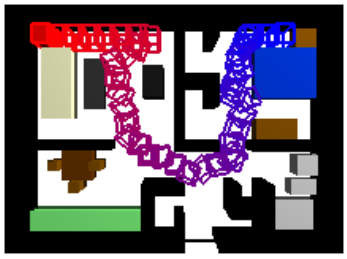
- In `Walls` [1] (Figure 4.11(d)), a simple 6 DOF stick-like robot must traverse a series of narrow passages (walls with holes) from one end of the environment to the other.

- In `Hook` (Figure 4.11(e)), an 8 DOF free-flying robot with three articulated links must maneuver through a wall with a small hole.

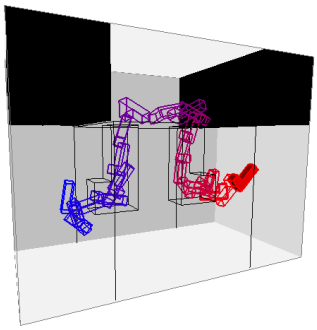
We only compare against I-RRT in the Heterogeneous environment because this is the only robot fully controllable by our interface, a mouse with 2 DOF.



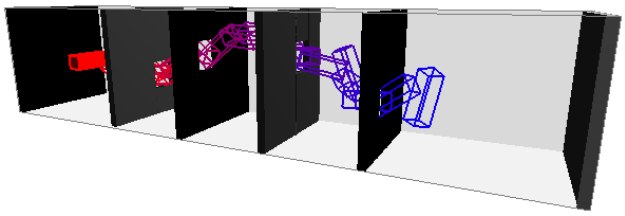
(a) `Heterogeneous` (2 DOF)



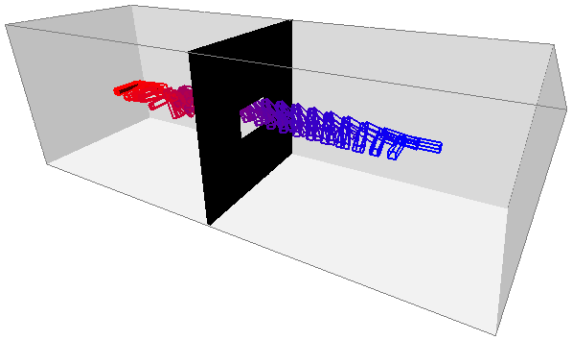
(b) `FloorPlan` (3 DOF)



(c) `LTunnel` (6 DOF)



(d) `Walls` (6 DOF)



(e) `Hook` (8 DOF)

Figure 4.11: Various environments for experimental analysis. All queries require traversal through narrow passages between the start (solid red) and goal (hollow blue) configurations.

Table 4.4: Success rates for the various PRMs in the test environments.

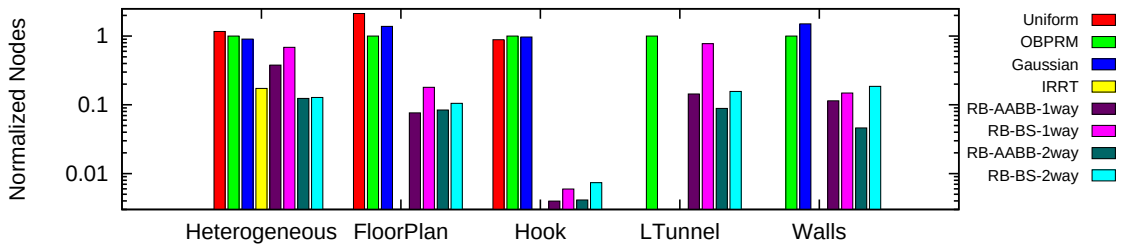
Planner	Heterogeneous	FloorPlan	Hook	LTunnel	Walls
Uniform	30%	50%	80%	0%	0%
OBPRM	70%	100%	100%	30%	100%
Gaussian	90%	80%	90%	0%	100%
I-RRT	100%	–	–	–	–
RB-AABB-1way	100%	100%	100%	100%	100%
RB-BS-1way	100%	100%	100%	20%	100%
RB-AABB-2way	100%	100%	100%	100%	100%
RB-BS-2way	100%	100%	100%	100%	100%

4.4.2 Results

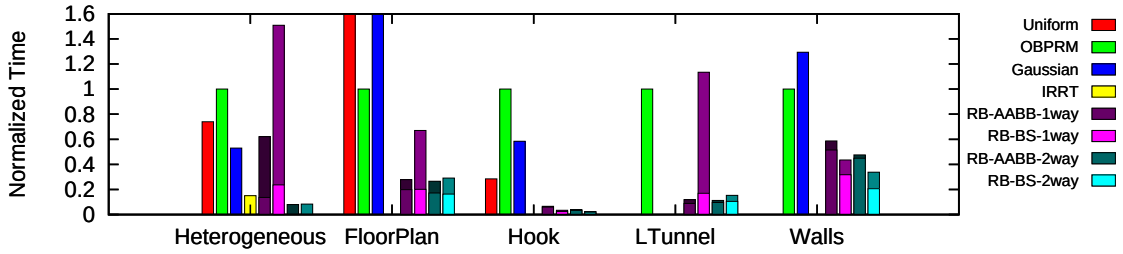
In our first experiment, we compare the mapping efficiency of Region-biased PRM with other PRMs. Table 4.4 shows the success rates of the various methods in the environments, Figure 4.12(a) displays the number of nodes in the final roadmap produced in each environment, and Figure 4.12(b) presents the time required by each method. In `FloorPlan`, Uniform and Gaussian had normalized times of 6.786 and 1.755, respectively, and were cut-off to better show the data.

4.4.2.1 Performance

Our experiments demonstrate that Region-biased PRM offers more reliable and efficient roadmap creation compared to the tested automatic methods. The user's input improves the number of successful construction attempts to 100% across all environments (in the intended two-way case). By examining the planner feedback, the user can identify workspace areas where the planner is unable to sample or connect nodes to the map, and then intervene by creating an attract region to bias sampling in those areas. This allows the collaborative strategy to focus system resources on difficult regions and provides greater robustness to sampling-based randomness compared to the automatic methods. Additionally, Region-biased PRM typically



(a) Nodes



(b) Time

Figure 4.12: (a) Number of nodes, and (b) time required by each method to solve the construction query, normalized to OBPRM. For the Region-biased PRM methods in (b), the upper portion of the bar represents the user's pre-specification time, while the lower portion represents the time taken by the automated planner after pre-specification.

improves construction efficiency in terms of both the number of nodes and the total time required to build the map (even with the overhead of collecting initial user input). Region-biased PRM's running time improved on the fastest automatic planners by a minimum of 46% in `Walls` and a maximum of 91% in `Hook`. I-RRT's performance is comparable to Region-biased PRM which performed slightly better in `Heterogeneous`. We also note that if the number of queries to solve were greater, the difference between the methods would be more pronounced as Region-biased PRM can reuse the computed roadmap. By taking advantage of the user's intuitive global analysis of the scene, Region-biased PRM can focus sampling in difficult areas between the connected components of a roadmap and, thus, achieve higher connectivity and reduced planning time. In turn, the reciprocal feedback given to the user,

including showing the roadmap and connected components, visualizing a region's usefulness, and recommending specific regions, can guide the user toward achieving these ends.

4.4.2.2 One- vs. Two-Way Communication

In three of the environments (`FloorPlan`, `Hook`, and `Walls`), the user strategies for one-way communication were very similar to their two-way counterparts. While the ability to correct input errors and delete unproductive regions contributed to performance, it was not the dominating factor in these cases. Conversely, the user strategies differed significantly in the other two environments (`Heterogeneous` and `LTunnel`). In these environments, the two-way strategies relied on the ability to modify the regions in order to map the space efficiently. For example, in `Heterogeneous` the user would typically begin with a large region in the center and modify it as the system provided feedback to make it smaller and more focused on areas that were not yet connected. This approach is not possible in one-way planning, and performance in that environment suffered from the inability to re-target the PRM's focus. In `LTunnel`, the two-way strategy for boxes achieved better performance than its one-way counterpart by simply deleting attract regions once a single connected component had broken through. The one-way spheres case for this environment was far more dramatic because it was too difficult for the user to precisely specify spherical regions that conservatively estimated the box-like tunnel. In the two-way case, the user could roughly estimate the regions required and then modify those that failed to contribute to the roadmap. The inability to make such adjustments prevented the user from building this map consistently with spherical regions. This implies that two-way interaction provides significant benefits when the workspace area of interest is shaped differently than the planning region.

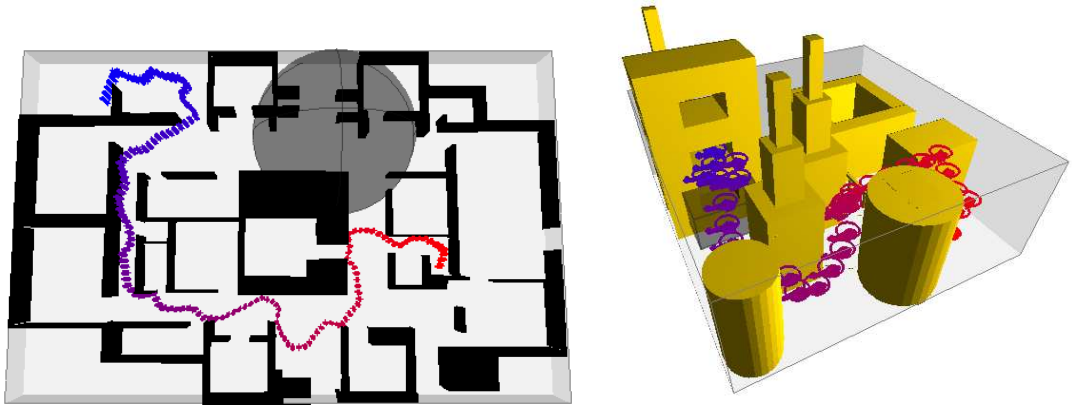
4.4.2.3 Region Shape

Our data shows that the user specification time for BS regions is generally less than that of AABB regions, but the planning time for BS regions is generally greater than for AABB regions. This suggests a trade-off between the ease of a region's manipulability and its effectiveness. However, the total mapping time does not seem to differ significantly. Furthermore, from our experience, different users prefer different region types depending on the environment and situation. While the one-way/two-way comparison hints that some of this disparity is related to how well a planning region fits the workspace area of interest, we leave the full investigation of these choices to a future user study. Additionally, in the future it would be interesting to study other region shapes that can be simply specified with a few user inputs. Specifically, Oriented Bounding Boxes (OBB) and ellipsoids would be of interest.

4.5 Experimental Analysis of Roadmap Customization

In this experiment, we illustrate roadmap customization through Region-biased PRM. The user is tasked with creation of a roadmap that avoids a specific area.

We test this in the two environments shown in Figure 4.13. `Building` is an office building in which several homotopically equivalent paths exist for a 2 DOF omni-directional robot. The avoidance region shown in dark gray represents some area of danger (such as a fire or collapsed portion of the building) that the robot should avoid. `Helicopter` is a cityscape that is traversed by a flying robot with 3 DOFs. In this environment, we require the robot to avoid flying through an open architecture of a building (again shown as a dark gray region). The construction query is designed so that there are at least two homotopically distinct paths from start to goal and at least one of them passes through the avoid region. We show the percentage of roadmaps in which the shortest path successfully avoids the region



(a) Building (b) Helicopter

Figure 4.13: (a) `Building` and (b) `Helicopter` environments used to illustrate roadmap customizability. Avoidance regions are shown in dark gray and queries are shown as solid red/hollow blue pairs.

Table 4.5: Percentage of maps with shortest paths correctly steering away from the avoidance regions.

Environment	PRM	Region-biased PRM
Building	20%	**100%**
Helicopter	50%	**100%**

for our Region-biased PRM compared to PRM. Ten trials were completed, and the successful percentages of 'safe' maps are shown in Table 4.5.

As we can see, our strategy is successfully able to avoid the regions that might be traversed by an automatically planned path. Additionally, the roadmaps created by Region-biased PRM contain no nodes in the avoid region, while the successful roadmaps created by PRM simply did not use their nodes in the avoid region for their shortest path. We would like to emphasize that although it may be possible to design these constraints into the problem specification or graph search, our strategy allows online customization during roadmap construction. This as-needed

specification makes Region-biased PRM well suited to handling newly discovered or temporary constraints without needing to alter the environment description. These simple tools enable a user to customize solutions for a variety of scenarios with minimal operational burden.

5. ON THE THEORY OF USER-GUIDED PLANNING

In this chapter, we extend the concept of (ϵ, α, β)–expansiveness [41] (Section 2.2.4) to theoretically explain when and by how much user input modalities (Chapter 3) perform better than uniform sampling alone.

5.1 Determining $\mathcal{C}_{free}$ Parameters

Before analyzing the models of user-guidance, we determine the ϵ and β for a given instance of $\mathcal{C}_{free}$. We characterize two important sets E^* and B^* that represent the critical configurations of $\mathcal{C}_{free}$ to sampling and connecting in sampling-based planning algorithms.

We note that these parameters and sets are as difficult to compute as the explicit boundaries of $\mathcal{C}_{obst}$. However, their purpose is twofold. First, this is the first work, to our knowledge, that attempts such descriptions of α and β that describe important notions of an instance of $\mathcal{C}_{free}$ — the hardest portions of the problem. Second, they provide a tool for analyzing the effect that specific (user) inputs have on planning with sampling-based motion planners.

5.1.1 ϵ

ϵ for a given $\mathcal{C}_{free}$ represents the ratio of volume of the smallest visibility set of any configuration to the volume of the entire space. Thus,

$$\epsilon = \min_{q \in \mathcal{C}_{free}} \frac{\mu(V(q))}{\mu(\mathcal{C}_{free})}$$

.

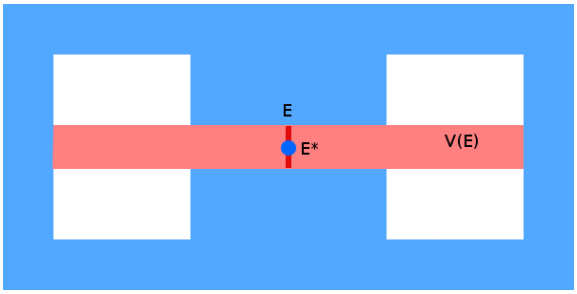


Figure 5.1: E (red line) is the set of configurations with the smallest visibility ratio. $V(E)$ is shown in transparent red. E^* is shown as a configuration in blue. Notice that E^* has an equivalent visibility to E.

Let E be the set of configurations with that smallest visibility ratio ϵ (Figure 5.1).

$$E = \left\{ q \in \mathcal{C}_{free} \middle| \frac{\mu(V(q))}{\mu(\mathcal{C}_{free})} = \epsilon \right\}$$

Let E^* be the smallest subset of E such that $V(E^*) \equiv V(E)$. In this way, E^* now represents a set of configurations with the lowest visibility in $\mathcal{C}_{free}$ (Figure 5.1).

5.1.2 β

Given a visibility set, there always exists a 0–LOOKOUT — the set of configurations the visibility set was based on. Instead, we seek the smallest maximal β–LOOKOUTs possible in $\mathcal{C}_{free}$ to represent the most difficult visibility sets to expand on or merge. We let

$$\beta(q) = \max_{q' \in V(q)} \frac{\mu(V(q') \setminus V(q))}{\mu(\mathcal{C}_{free} \setminus V(q))}$$

be a function to compute the largest β for a visibility set around a configuration q. Then,

$$\beta = \min_{q \in \mathcal{C}_{free}} \beta(q)$$

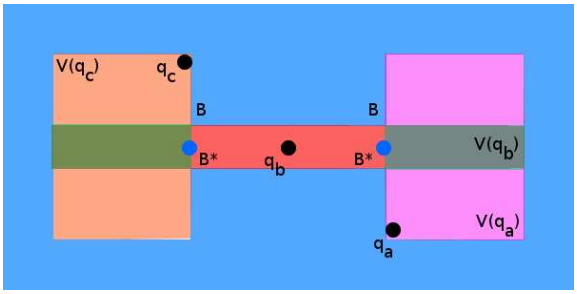


Figure 5.2: B (green line) is the set of configurations that are part of the smallest maximal β–LOOKOUTs across all of $\mathcal{C}_{free}$. Configurations q_a, q_b, q_c are representative configurations that have a small maximal β–LOOKOUT, their visibility sets are shown in transparent red. B^* is shown in blue. Notice that B^* has an equivalent visibility to B.

.

Let B be the set of configurations that are part of the smallest maximal β–LOOKOUTs across all of $\mathcal{C}_{free}$ (Figure 5.2).

$$B = \{q \in \mathcal{C}_{free} | \quad \exists q' \in \mathcal{C}_{free} : \beta(q') = \beta \wedge q \in \beta\text{–LOOKOUT}(V(q'))\}$$

Let B^* be the smallest subset of B such that $V(B^*) \equiv V(B)$ (Figure 5.2). B^* represents a set of configurations that are the most difficult to expand a roadmap to include, i.e., represent the configurations to which it is the hardest to connect.

Because the β–LOOKOUT might not have any volume, α can be zero, therefore we do not analyze any effects on this parameter for any user-model. Recall that α and β correlate to the probability of sampling one node in the visible area of a roadmap and one in the complement of the roadmap's visibility. In our formulation of β, a single configuration may be chosen instead of sampled, which both connects to the roadmap and sees a maximal portion of the complement to the roadmap's visibility. In other words, the one configuration is an optimal selection to serve the same purpose as a node both in the visible region of a roadmap and in its complement, thus expanding

the roadmap's visibility and connectivity simultaneously.

5.2 Analysis of User-guided Planning

In this section, we explore a novel extension of (ϵ, α, β)–expansiveness, which we call *augmented (ϵ', α', β')–expansiveness*. At the core of this concept is considering (ϵ, α, β)–expansiveness of an augmented $\mathcal{C}_{free}$, namely $\mathcal{C}_{free}$ with configuration "beacons" located at the points of lowest visibility or most difficult portions of the space to connect. When placed properly, the "beacons" allow derivation of augmented (ϵ', α', β')–expansiveness parameters that are greater than the inherent (ϵ, α, β)–expansiveness of $\mathcal{C}_{free}$ implying the motion planning problem is easier to solve. In the following models, we explore various user inputs using this augmentation of $\mathcal{C}_{free}$. When our assumptions of the user input hold the problem becomes easier to solve. Finally, we examine open questions and discuss aspects of the models for future exploration. Experimental support of the proposed theory can be found in Section 3.2.

5.2.1 Configuration-based Input

First, we assume an idealized model of input, where a user places specific configurations in $\mathcal{C}_{free}$ to act as waypoints for the sampling-based motion planner, e.g., [45] (Section 3.1.1). We call these augmenting configurations "beacons" and analyze the augmented (ϵ', α', β')–expansiveness parameters of $\mathcal{C}_{free}$.

Lemma 5.2.1. *When a user provides a set of configurations R as input to a sampling-based planner such that $V(R) \supseteq V(E^* \cup B^*)$ (Figure 5.3), the augmented (ϵ', α', β')–expansiveness parameters of $\mathcal{C}_{free}$ are greater than the inherent (ϵ, α, β)–expansiveness parameters of $\mathcal{C}_{free}$.*

Proof. ϵ'–Since $V(E^*) \subseteq V(R)$, the set of configurations R provides coverage to the

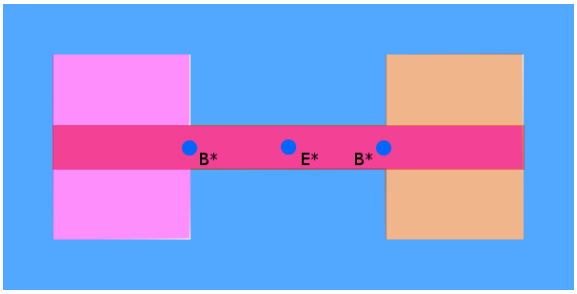


Figure 5.3: Configurations placed on E^* and B^* that represent configuration-based input that can help a sampling-based motion planner. The visibility regions of each configuration are shown, notice how they act as "beacons" for the space making the problem "easier."

set of lowest visibility configurations in $\mathcal{C}_{free}$, and by definition there are no lower visibility configurations in $\mathcal{C}_{free} \setminus V(R)$. Thus, we have

$$\epsilon' = \min_{q \in \mathcal{C}_{free} \setminus V(R)} \frac{\mu(V(q))}{\mu(\mathcal{C}_{free})} > \epsilon.$$

β'–Since $V(B^*) \subseteq V(R)$, the set of configurations R represents regions of $\mathcal{C}_{free}$ that are most difficult to connect to, and by definition there will be no configurations more difficult to connect left in $\mathcal{C}_{free} \setminus V(R)$. Thus, we have

$$\beta' = \min_{q \in \mathcal{C}_{free} \setminus V(R)} \beta(q) > \beta.$$

α'–Based on our definition of β and β' and the fact that β is not decreased, we have $\alpha' \geq \alpha$. □

We note that because of the precision required related to gathering this information, this methodology of user-guidance may not be ideal. Hence, a goal of this theory is to explore the effect of this guidance. As such, we have shown that this input modality clearly solved the problem. We can extend the analysis to better un-

derstand the improvement gained from augmented (ϵ', α', β')–expansiveness, which is expected to simply be an improved convergence rate in the probability of success as the number of samples increases. Specifically, Corollary 5.2.2 shows that the expected number of random samples in the augmented space is a fraction of the expected number in $\mathcal{C}_{free}$ to generate a connected roadmap.

Corollary 5.2.2. *Let $\gamma \in (0,1)$ be a constant. Let $\delta_\epsilon = \epsilon'/\epsilon$, $\delta_\beta = \beta'/\beta$, and $\delta_\epsilon = \alpha'/\alpha$. Let $n = \lceil 16\ln(8/\epsilon\alpha\gamma)/\epsilon\alpha + 6/\beta + 2\rceil$ be the size of a roadmap in $\mathcal{C}_{free}$. When a user provides a set of configurations R as input to a sampling-based planner such that $V(R) \supseteq V(E^* \cup B^*)$, a roadmap generated in the augmented space of size $O(\frac{1}{\delta_\beta} - \frac{\ln \delta_\epsilon \delta_\alpha}{\delta_\epsilon \delta_\alpha})$-ratio of n will be connected with probability $1-\gamma$.*

Proof. In [42], it was shown that a roadmap of size $n = \lceil 16\ln(8/\epsilon\alpha\gamma)/\epsilon\alpha + 6/\beta + 2\rceil$ will be connected with probability $1-\gamma$. We analyze the ratio n'/n where n' is the number of nodes required to elicit a connected roadmap with probability $1-\gamma$ in the augmented space. Based on [42], $n' = \lceil 16\ln(8/\epsilon'\alpha'\gamma)/\epsilon'\alpha' + 6/\beta' + 4\rceil$. We substitute δ_ϵ, δ_β, and δ_α to find $n'/n = O(\frac{1}{\delta_\beta} - \frac{\ln \delta_\epsilon \delta_\alpha}{\delta_\epsilon \delta_\alpha})$. □

In summary, Lemma 5.2.1 analyzed augmented (ϵ', α', β')–expansiveness parameters to show they are improved over the inherent parameters of $\mathcal{C}_{free}$, and Corollary 5.2.2 showed that fewer random samples are expected to be required to generate a connected roadmap covering $\mathcal{C}_{free}$.

5.2.2 Path-based Input

In this model, the user provides a set of paths Π through the narrow passages of $\mathcal{C}_{free}$, essentially so that $V(\Pi) \supseteq V$ for some visibility region (Section 3.1.2). In this model, we assume that the end points of each path are added as nodes to the roadmap, and the path is then added as an edge in the same graph.

Lemma 5.2.3. *When a user specifies a set of paths* Π *such that* $V(\Pi) \supseteq V(E^* \cup B^*)$*, the augmented* ($\epsilon'$, α', β')*–expansiveness parameters of* $\mathcal{C}_{free}$ *are greater than the inherent* (ϵ, α, β)*–expansiveness parameters of* $\mathcal{C}_{free}$*.*

Proof. Since, Π is a path contained within $\mathcal{C}_{free}$ that encapsulates all the difficult portions of the problem, then based on similar logic to the proof of Lemma 5.2.1, the augmented (ϵ', α', β')–expansiveness parameters are greater than the inherent (ϵ, α, β)–expansiveness parameters. □

5.2.3 Region-based Input

Here, we extend the theory to approximate $\mathcal{C}_{space}$ regions (input model as described in Section 3.1.3). We assume that given a visibility region V^* the user can provide regions R as input in $\mathcal{C}_{space}$ such that $V^* \subseteq R \subset \mathcal{C}_{space}$ and $R \cap \mathcal{C}_{free} \subset \mathcal{C}_{free}$. Without loss of generality, we let the planner have a probability $p > 0$ to select a sample from R and probability $1 - p$ to sample from the entire space — this is to maintain probabilistic completeness of the automated technique. Here, augmented (ϵ', α', β')–expansiveness is derived not from configuration "beacons" but from a subset of $\mathcal{C}_{free}$.

Lemma 5.2.4. *When a user specifies regions* R *such that* $R \cap \mathcal{C}_{free} \subset \mathcal{C}_{free}$ *and* $R \supseteq V(E^* \cup B^*)$*, the augmented* ($\epsilon'$, α', β')*–expansiveness parameters of* $\mathcal{C}_{free}$ *are greater than the inherent* (ϵ, α, β)*–expansiveness parameters of* $\mathcal{C}_{free}$*.*

Proof. First, we analyze ϵ'. We define $\epsilon_R = \min_{q \in R_{free}} \frac{\mu(V(q))}{\mu(R_{free})}$ where $R_{free} = R \cap C_{free}$. Then based on the probability of sampling, the overall $\epsilon' = (1-p)\epsilon + p\epsilon_R$ which is surely greater than ϵ because ϵ_R is strictly greater than ϵ.

Second, we analyze β'. We define $\beta_R = \min_{q \in \mathcal{C}_{free}} \beta(q)$. Then based on the probability of sampling, the overall $\beta' = (1-p)\beta + p\beta_R$ which is surely greater than β based on similar logic as before. □

In summary, we presented a novel description of the inherent parameters of $\mathcal{C}_{free}$ and described conditions that various user input methods aid sampling-based planners through a novel concept called augmented (ϵ', α', β')–expansiveness.

6. KINODYNAMIC REGION-BIASED RRT

In this chapter, we present a region-biased kinodynamic planner to handle non-holonomic systems, which are complex systems planning under velocity and acceleration constraints, e.g. a car-like robot.

6.1 State Space

When considering kinodynamic constraints, forces must be applied to a robot in order for a robot to move. Because of this, $\mathcal{C}_{space}$ is not a sufficient abstraction for these types of robots. Instead, we consider planning in State Space ($\mathcal{X}_{space}$). Here a state of our robot will include all of the positional DOFs along with velocities of each DOF. Note that this doubles the dimensionality of the motion planning problem.

At each time-step, an action, or control, can move a robot state to a new one. This is equivalent to applying a force and/or torque to the robot for an amount of time.

6.2 Algorithmic Modifications

RRTs are well suited for motion planning in $\mathcal{X}_{space}$. Here, we apply our region-based framework to an algorithm called Kinodynamic RRT. Our algorithm, a modification of Region-biased RRT, is shown in Algorithm 7. We will keep referring to this approach as Region-biased RRT for simplicity.

The algorithm first selects a random region r for the current iteration. Note here, that the entire $\mathcal{X}_{space}$ is defined as an implicit region in order to maintain probabilistic completeness of RRT, and each region has an equal probability to be selected. From r a random position is sampled to seed the position values of x_{rand} while the rest of the state (rotations and velocities) are randomly sampled like a typical RRT. In this

Algorithm 7 Kinodynamic Region-biased RRT

Input: An Environment e, a maximum step-size Δ
Output: Tree T

```
1: T ← ∅
2: while ¬done do
3:    r ← SelectRegion(e.regions)
4:    x_rand ← r.GetRandomState()
5:    x_near ← NearestNeighbor(T, x_rand)
6:    x_new ← Steer(x_near, x_rand, Δ)
7:    T.Update(x_near, x_new)
8:    UpdateDisplay(T, e)
9: return T
```

way, the 2- or 3- dimensional workspace region can bias random samples in $\mathcal{X}_{space}$.

Then, the algorithm proceeds as a typical RRT. the nearest node in the tree x_{near} is determined, and a node x_{new} is generated by extending from x_{near} towards x_{rand}. Because the robot has kinodynamic constraints, in order to steer towards a new state, we must select a control and integrate over a time step Δt for the Steer function. The original kinodynamic RRT [61] presented two heuristics each for selecting a control and time step. For the control, we can select one of the available controls either randomly or based on the resulting distance to x_{rand}. For the time step, we can choose either a fixed or variable length time over which to apply the control. We support all four of the control selection/integration time combinations. We could also use an exact steering function if one is available for the specific robotic system. After finding x_{new}, the tree is updated by adding a new node x_{new} and the edge $\{x_{near}, x_{new}\}$ to it. Essentially, the only change to our region-based collaboration framework when considering kinodynamic constraints is a change to the underlying planner.

If a user defines any avoid regions (essentially a virtual obstacle not known *a priori* to planning), RRT extensions are not permitted to enter these regions.

Finally, the planner will communicate the progress to the user through updating the display of regions and the tree.

6.3 Experimental Analysis

In this section, we compare our collaborative planner against its unguided, fully automated counterpart in two scenarios. Because our region-based guidance can be used in conjunction with many RRT variants, we chose to evaluate the proof-of-concept by testing guidance on the canonical Kinodynamic RRT.

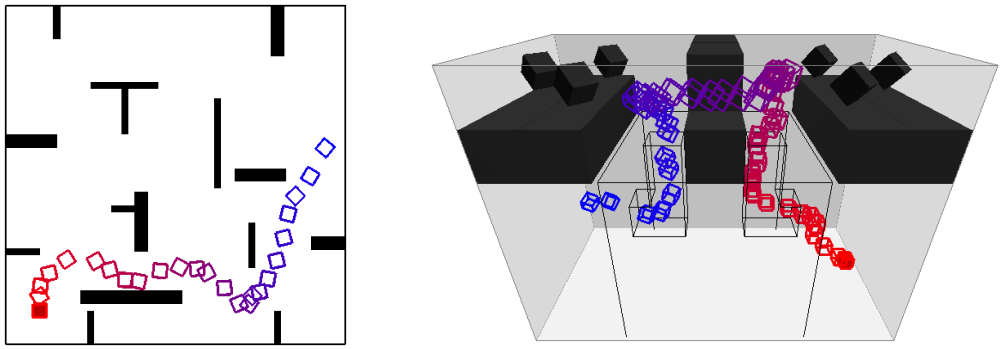
6.3.1 Setup

To evaluate our methods, we compare the running time of our Region-biased RRT with its unguided counterpart in solving the single-query scenarios seen in Figure 6.1. Queries are defined as a start configuration (red) and goal configuration (blue) pair.

- In `Planar` (Figure 6.1(a)), a non-holonomic robot with 3 positional DOF and 4 discrete controls must steer around a set of right-angle barriers.
- In `Tunnel` (Figure 6.1(b)), a non-holonomic robot with 6 positional DOF and 12 discrete controls must traverse a narrow tunnel.

Experiments consisted of 10 trials. Execution continued until the query was solved or the tree grew to 15,000 nodes. We evaluate the number of successful completions and the average time for all executions (including user input and collaboration time).

The user-guided executions were performed by graduate and high school students studying motion planning. To minimize the impact of user variance, the users were allowed to practice with the system until they developed consistent performance. Recall, as mentioned in Chapter 4, this research is focused on the usefulness of user collected information and not on the particular interface.



(a) `Planar` (6 DOF) (b) `Tunnel` (12 DOF)

Figure 6.1: Example scenarios used in experimental analysis. (a) A non-holonomic robot with 3 positional DOFs in a planar environment (6-dimensional state space). (b) A non-holonomic robot with 6 positional DOFs in a volumetric environment (12-dimensional state space). All queries require traversal through narrow passages between the start (solid red) and goal (hollow blue) configurations.

6.3.2 Analysis

We compared our Region-biased RRT with a standard kinodynamic RRT (KRRT) in the `Planar` and `Tunnel` environments. We used a weighted Euclidean distance function, fourth order Runge-Kutta (RK) integration, an integration time-step of 0.01, and a maximum RRT time-step of one second. For both of these approaches, we select our control based on a variable time step and best control selection as described in [61]. Essentially, we pick a random time-step up to one second, attempt all controls, and select the one which is closest to the growth node x_{rand}.

The robots in these scenarios use discrete rather than continuous controls. In `Planar`, the robot can apply a linear force to accelerate forward or backward or apply an angular force to spin left or right. In `Tunnel`, the system is fully actuated, as in, a forward and backward force can modify each degree of freedom separately. There is no friction analog to disperse momentum in either scenario, so the robots must apply retrograde forces to reduce velocity.

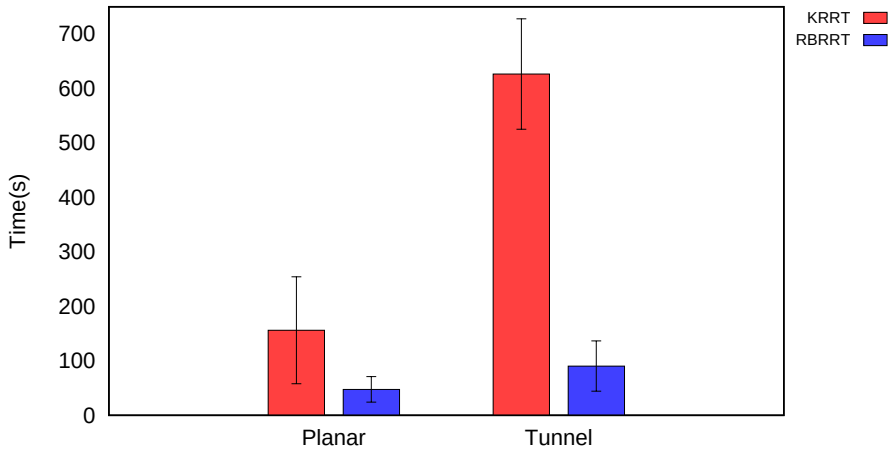


Figure 6.2: Average running times and standard deviations for non-holonomic experiments. For `Planar`, Region-biased RRT exhibits reduced mean planning time with a p-value of 0.0064 for a one-tailed t-test. For `Tunnel`, the difference is more dramatic with a p-value of 0.0001.

Table 6.1: Non-holonomic scenario success rates.

	`Planar`	`Tunnel`
KRRT	50%	10%
Region-biased RRT	**100%**	**100%**

Results, shown in Figure 6.2 and Table 6.1, demonstrate that user guidance significantly reduced the mean and variance of planning time and improved success rates in both cases. For `Planar`, Region-biased RRT exhibits reduced mean planning time with a p-value of 0.0064 for a one-tailed t-test. For `Tunnel`, the difference is more dramatic with a p-value of 0.0001.

The user's strategy was to create a single region and slowly drag it through the environment as a way to lead the RRT around obstacles. This was expected because the extensions that cause the robot to build forward velocity tend to travel further than those that don't. Thus, the configurations nearest to a given x_{rand} (sampled

from within a region) tended to have larger momentum and therefore required more control applications to change course.

Another point of interest is that the user-guided method shows less variation in running time than the fully automatic method. After some initial exploration of the easily reachable areas, the majority of new nodes tended to follow the user's guidance. This had a stabilizing effect on performance because the user's strategy essentially dominated the planner's general progress and because the users tended to recycle successful strategies. Thus, the user-guided planner focused its resources on finding a plan in the same homotopy class across each execution, whereas the fully automated method had no such consistency.

7. DYNAMIC REGION-BIASED RRT

The motion planning problems in many of the environments discussed in this dissertation, and in many problems in robotics, are strongly related to the workspace. That is to say, the solution paths of the motion planning problem are highly similar to the paths in the workspace. A user can provide great intuition in these problems as we have shown. However, there is hope for some types of environments that we can create a fully automated approach that can mimic the human's input. By doing this, we can eliminate the time required to gather information from a user and the time needed to render the scene and provide feedback to the user. So, we can trade the full power of human intuition for a more efficient automated approach. In the future, one could explore synergies between the two approaches.

In this chapter, we explore a fully automated collaborative algorithm that was inspired by the human collaboration techniques of Chapter 4. The idea is to compute a graph embedding in the workspace that can be used to automatically guide the regions as a planner progresses. We specifically target an RRT approach as it was most natural for this approach.

7.1 Algorithm

Our methodology to automate region-based collaboration relies on a spatially embedded graph in the workspace which will bias a planner in a similar way to Region-biased RRT. In this section, we specifically outline the general methodology and define the data structures used and the requirements they must possess.

Our algorithm, Dynamic Region-biased RRT, Algorithm 8, first begins with a pre-computation step in which the workspace is processed and a spatially embedded graph, called the *Embedding Graph* (Definition 7.1.1), is computed. Then, when a

Algorithm 8 Dynamic Region-biased RRT

Input: Environment e and a Query $\{q_s, q_g\}$
Output: Tree T

```
1: G ← ComputeEmbeddingGraph(e)
2: F ← ComputeFlowGraph(G, q_s)
3: T ← (∅, ∅)
4: R ← InitialRegions(F, q_s)
5: while ¬done do
6:    RegionBiasedRRTGrowth(T, F, R)
7: return T
```

query is given, we compute a directed version of the embedding graph, which we call the *Flow Graph* (Definition 7.1.2), which encodes the directions of exploration throughout the environment. Last, as discussed in a following section, our planner will dynamically create, modify, and destroy regions to bias RRT growth in $\mathcal{C}_{space}$. This approach is inspired by the human collaboration found in Region-biased RRT, and attempts to mimic it.

7.1.1 Embedding Graph

Dynamic Region-biased RRT begins by pre-computing an *Embedding Graph* (Definition 7.1.1) of the free workspace. It is a one dimensional, spatially embedded skeleton of the workspace.

Definition 7.1.1. *An **Embedding Graph**, $G = (V, E)$, is a one dimensional, spatially embedded skeleton of the workspace (undirected graph). The skeleton must be a retraction, i.e., every point in the free workspace can be mapped onto the skeleton. The vertices are points, and the edges, also referred to as arcs, are polygonal chains through the workspace.*

The embedding graph has certain requirements to maintain desirable properties. Namely, the embedding graph must be a deformation retract [34] of the free

workspace. This implies that every point in the free workspace can be mapped onto the skeleton, i.e., the embedding graph encodes the topology of the workspace. This property is quite desirable in motion planning because we can infer and approximate the total number of possible paths for a motion planning problem. In the future, we plan to relax these properties through an approximation of a true embedding graph and analyze how it affects our planner.

There are a few possible data structures that satisfy this property. One, a Generalized Voronoi Graph (GVG) [18] is the set of points equidistant to m obstacles in a space of dimension m. The GVG, however, is not always guaranteed to be connected. A few methods have proposed ideas to overcome this [18, 31]. Another possibility is a Reeb Graph [80] that represents transitions in level sets of a real-valued function on a manifold, i.e., nodes of the graph are critical values of the function, referred to as a Morse function, and edges are the topological transitions between them. It has applications in various parts of computational geometry and computer graphics, e.g., shape matching [37], iso-surface remeshing [101], and simplification [100].

In this work, we introduce a novel algorithm for computing an embedding graph. Our algorithm computes and spatially embeds a Reeb Graph in the free workspace. We do not claim that this algorithm is optimal in any sense, but it is sufficiently simple to compute a Reeb Graph that satisfies our requirements of an embedding graph.

Our algorithm, Algorithm 9, begins by computing a Delaunay tetrahedralization of the free workspace [88]. Using the 2-skeleton of the computed tetrahedralization to initialize a Reeb Graph, we use the triangles of the tetrahedrons to inform and reduce the Reeb Graph through the algorithm found in [77]. We then embed the Reeb graph into the free workspace by shifting each node of the Reeb Graph to its closest tetrahedron. Then, for each arc of the Reeb Graph we find a path through the

Algorithm 9 Compute Embedding Graph

Input: Environment e
Output: Embedding Graph G

```
1: D ← Tetrahedralization(e)
2: R = (R_V, R_E) ← ConstructReebGraph(D)
3: G = (V, E) ← (∅, ∅)
4: for all v ∈ R_V do
5:    V ← V ∪ {D.ClosestTetrahedron(v)}
6: for all e ∈ R_E do
7:    s ← e.Source(); t ← e.Target()
8:    E ← E ∪ (s, t, D.FindPath(s, t))
9: return G
```

dual of the tetrahedralization between nodes. We specifically bias the path search to be constrained to the tetrahedrons related to that specific Reeb arc. We know this mapping because we associate tetrahedrons with each edge of the 2-skeleton used to initialize the Reeb edges, and then as arcs are merged, we merge the associated sets of tetrahedrons. In the path construction, we also choose paths which lie entirely in the free workspace by always going through the adjacent triangle between two tetrahedrons instead of using straight lines between tetrahedrons. In this way, the final embedding graph is a set of points and polygonal chains connecting these points that are spatially contained in the free workspace.

7.1.2 Flow Graph

Once the Embedding Graph is computed and a query is requested to be solved, our planning algorithm computes a *Flow Graph* (Definition 7.1.2) of the environment that will be used in the coordination of region construction, modification, and deletion. We define a flow graph as a directed embedding graph.

Definition 7.1.2. *A **Flow Graph**, $F = (V, E)$ is a directed embedding graph.*

To compute the flow graph, we first find the closest embedding graph vertex

to the start configuration. Then, we compute our flow with a breadth-first search traversal of the embedding graph. We include cross-edges in the final directed graph.

7.1.3 Region-biased RRT Growth

Our RRT growth strategy is shown in Algorithm 10. The algorithm takes in an RRT tree T, the flow graph F, and the set of regions R, and is broken into four phases. In the first phase, the algorithm acts like Region-biased RRT (Section 4.2.2). We select a region uniformly at random and select a random configuration from it — note that the entire environment is still considered a region to maintain probabilistic completeness just like the collaborative algorithms in Chapter 4. In the second phase, we advance the selected region r along its associated flow graph edge. In order to do this, we simply determine if the extended node q_{new} reached some portion of r. If so, we move r to the next point on the embedded flow edge in the workspace. The third phase will delete old and useless regions determined by some threshold τ of failure. Finally, we will spark new regions if q_{new} comes within an ϵ distance of an unexplored flow vertex v. We create a region for each outgoing edge of v and mark v as explored in the flow graph.

Our algorithm has little overhead compared with standard RRT growth. It comes in the form of selecting, advancing, creating, and deleting regions which all occurs in amortized constant time.

7.1.4 Example

Figure 7.1 shows a model execution of our approach. First, given an environment, shown in Figure 7.1(a), we compute an embedding graph (magenta) representing the possible routes of exploration in the workspace, shown in Figure 7.1(b). Then, at query time, we compute a flow graph (magenta), shown in Figure 7.1(c), to guide the exploration of the collaborative planning algorithm. From here, we initialize a set

Algorithm 10 Region-biased RRTGrowth

Input: Tree T, Flow Graph F, Regions R

Require: ϵ is a multiple of the robot radius, τ is a maximum for failed extension

{Region-biased RRT extension}

1: $r \leftarrow$ SELECTREGION(R)
2: $q_{rand} \leftarrow r$.GETRANDOMCFG()
3: $q_{near} \leftarrow$ NEARESTNEIGHBOR(T, q_{rand})
4: $q_{new} \leftarrow$ EXTEND($q_{near}, q_{rand}, \Delta$)

{Advance region along Flow edge}

5: **while** r.INREGION(q_{new}) **do**
6: r.ADVANCEALONGFLOWEDGE()
7: **if** r.ATENDOFFLOWEDGE() **then**
8: $R \leftarrow R \setminus \{r\}$

{Delete useless regions}

9: **if** r.NUMFAILURES() $> \tau$ **then**
10: $R \leftarrow R \setminus \{r\}$

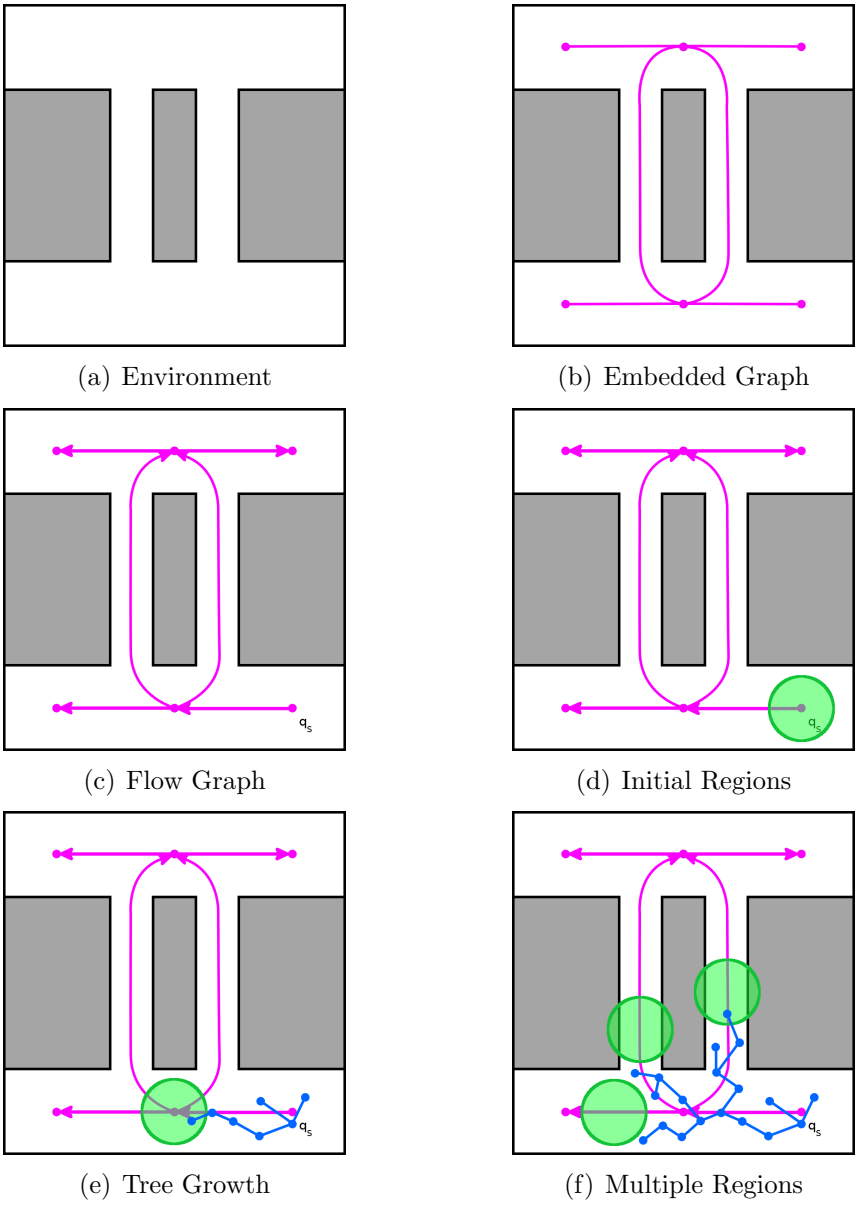
{Create new regions}

11: **for all** $v \in F$.UNEXPLOREDVERTICES() **do**
12: **if** $\delta(v, q_{new}) < \epsilon$ **then**
13: $R \leftarrow R \cup$ NEWREGION(v)
14: F.MARKEXPLORED(v)

of regions (green), which in this case is only one region, from the start node of the embedding graph to begin exploring the space, Figure 7.1(d). Figure 7.1(e) shows how the regions bias tree growth (blue) similar to our collaborative Region-biased RRT algorithm. As the algorithm proceeds and reaches the next node of the flow graph, three more regions are created to traverse the outgoing edges from that node, Figure 7.1(f). The algorithm proceeds in this manner until a stopping criteria is met, e.g., query is solved or maximum number of iterations is reached.

7.2 Experimental Analysis

In this section, we highlight current results of Dynamic Region-biased RRT compared against RRT. There is still room for performance improvement which we include in the discussion.



(a) Environment (b) Embedded Graph

(c) Flow Graph (d) Initial Regions

(e) Tree Growth (f) Multiple Regions

Figure 7.1: Example execution of Dynamic Region-biased RRT: (a) environment; (b) precomputed embedding graph (magenta) in workspace; (c) flow graph (magenta) computed from the start position; (d) initial region (green) placed at the source vertex of the flow graph; (e) Region-biased RRT growth (blue); and (f) multiple active regions (green) guiding the tree (blue) among multiple embedded arcs of the flow graph (magenta).

7.2.1 Setup

All methods were implemented in a C++ motion planning library developed in the Parasol Lab at Texas A&M University. It uses a distributed graph data structure from the Standard Template Adaptive Parallel Library (STAPL) [12], a C++ library designed for parallel computing.

All experiments were run on a Dell Optiplex 9010 running Fedora 20 with an Intel(R) Core(TM) i7-3770 CPU with 24 GB of RAM with the GNU gcc compiler version 4.8.3.

We compared Dynamic Region-biased RRT and RRT [61] in the `MazeTunnel` environment, shown in Figure 7.2. This environment was particularly chosen because it contains narrow passages in the workspace which do not exist in $\mathcal{C}_{space}$. Our method will detect and attempt to plan on all these paths, but will dynamically adjust away from the false passages accordingly. We use $\Delta q = 10$ which is approximately 10% of the diagonal of each environment.

We compute total on-line planning time in seconds averaged over 10 trials. Outliers are removed using standard statistical techniques. Averages and standard deviations are shown in Figure 7.3.

7.2.2 Discussion

As Figure 7.3 shows, our planner exhibits faster online planning time as compared with RRT. The results are significant at $\alpha = 0.05$ with a p-value of 0.0358 using a student's t-test. In terms of pre-computation time, our embedding graph took on average 30 seconds to build. It took a little over half a second to compute a tetrahedralization of this complex environment with 2388 triangles into 44,086 tetrahedrons. It took about 29 seconds to reduce the tetrahedralization to a Reeb Graph of 26 nodes and 27 edges and negligible time to perform the edge embedding.

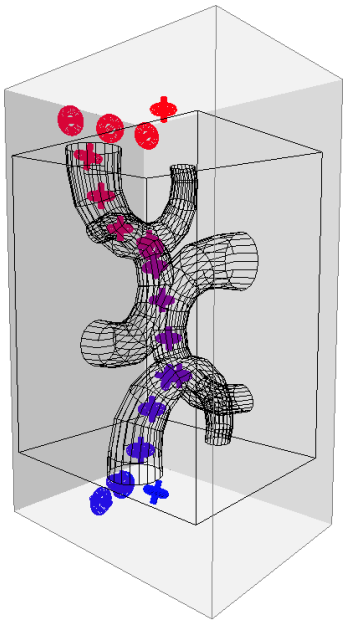


Figure 7.2: `MazeTunnel` environment. Note how there are false passages in the workspace in which the robot cannot pass through.

Interestingly, we can clearly see the benefit when we have the embedding graph, however if there will only be one query performed in an environment the pre-computation might not be worth it. It is important to note that the Reeb Graph computational efficiency is $O(n^2)$ in the worst case, where n is the number of points in the tetrahedralization [77]. It is theoretically possible to reduce this computation time to a little worse than $O(n \log n)$ with a more sophisticated construction algorithm [29]. In other words, if the environment was simpler the Reeb Graph computation would be more efficient.

Beyond this, we would like to make a special note of the dynamic aspect of our regions. In this test environment, there are false passages in the workspace. Our algorithm will create a region for those edges, which in turn distracts the planner. However, if enough failures occur, we will delete the region as it is likely a false passage and the region was not useful. In this way, our algorithm is robust to

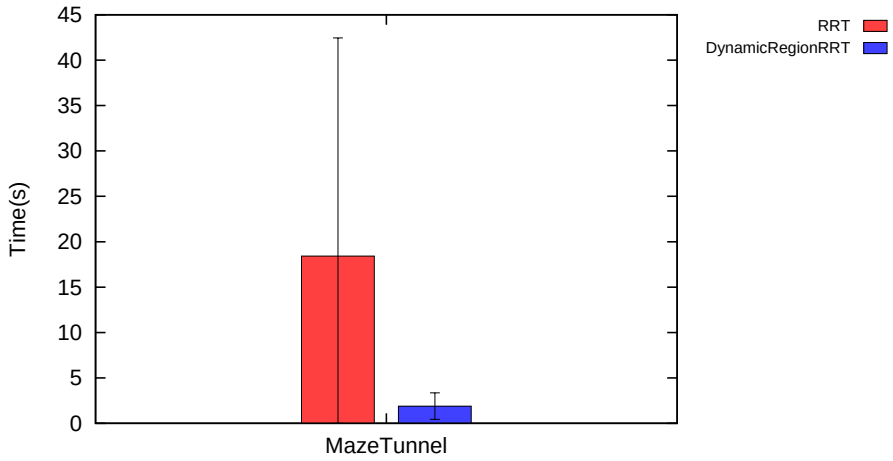


Figure 7.3: On-line planning time comparing Dynamic Region-biased RRT with RRT.

complex environments with multiple homotopy classes in the workspace. This is one of the unique aspects to our workspace-biased approach compared with similar approaches. In the future, we believe that we can effectively extend this approach for planning among dynamically changing environments by modifying the embedding graph during runtime to adjust for the movement of the obstacles.

8. CONCLUSION

In this work, we developed and analyzed a novel collaboration framework that captures human intuition and uses it to bias a sampling-based motion planner. We ensured our framework could be applied to many different sampling-based planners, applicable to complex robot systems, and even used these ideas to create a new fully automated approach to motion planning. Through this research, we saw the immense benefit of collaboration. Collaboration can have impact in various domains such as robotics and virtual prototyping, which we had in mind when developing our methodologies.

Specifically, we classified, modeled, and compared common approaches to user-guided motion planning to motivate further study into region-based approaches (Chapter 3). From here, we presented a region-based framework whereby a user can specify workspace regions to attract or repel a sampling-based motion planner (Chapter 4). We showed variants of our framework for graph-based, tree-based, and hybrid methods. In Chapter 5, we revisited the various user input modalities to analyze the approaches theoretically. Finally, we showed that the framework can be extended in a few interesting directions. First, it is able to handle kinodynamic constraints and plan motions for non-holonomic robotic systems (Chapter 6). Second, we showed a human-inspired approach for automatically moving regions through an environment (Chapter 7). We showed how low-dimensional workspace input can effectively bias a high dimensional motion search in a variety of ways. By keeping a simple interface for a user, we are able to provide a collaboration framework for many types of robots and sampling-based planners. This framework also has an added benefit of allowing intermittent user intervention into the planning process.

Through our theory and discussion of user inputs, we motivate the need to further explore a few directions. Specifically, the effectiveness of the user-input can inform interface designers for human-robot interaction and virtual prototyping. Through these applications the trade-off of user input time vs planning effectiveness can be fully explored. Additionally, we motivate future advanced heuristics in planning. As an example, two automated planners might be able to work together through region input or path based input instead of only working with singular configurations. In fact, a few examples have been seen of this [73, 98, 79], but more research is needed in this direction.

Additionally in the future, we would like to extend our region-based framework in several ways. First, we believe that we can generalize the biasing of the method to not only set the center of mass of the robot, but really any point of interest on the robot. This would allow a user to influence planners in difficult scenarios where rotational degrees of freedom might dominate, e.g., manipulator arms. In this case, we can use the regions to bias the position of the end effector and let the planner determine the remaining DOFs. Second, we can see the approach extended to control motions for many agents cooperating for use in gaming and virtual reality settings. Our framework could provide a general path the agents should follow in the workspace, while the underlying planner utilizes reactive behaviors within the agent group to derive precise motions. Last, we would like to incorporate this framework into computer-aided design scenarios by studying its applicability to virtual prototyping. Specifically, a human operator can use our framework to plan a motion to remove a part from a complex assembly. Then, by analyzing the part's motions, the product by extension can be validated in terms of maintainability.

REFERENCES

[1] N. M. Amato. Motion planning benchmarks. `http://parasol.tamu.edu/groups/amatogroup/benchmarks/`. Accessed: June 1, 2015.

[2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. OBPRM: an obstacle-based PRM for 3d workspaces. In *Algorithmic Foundations of Robotics III*, pages 155–168, Natick, MA, USA, 1998. A. K. Peters, Ltd.

[3] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. Robot. Automat.*, 16(4):442–447, August 2000.

[4] N. M. Amato and G. Song. Using motion planning to study protein folding pathways. *J. Comput. Biol.*, 9(2):149–168, 2002.

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

[6] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, May 2009.

[7] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Roadmap-based flocking for complex environments. In *Proc. Pacific Graphics*, pages 104–113, Oct 2002.

[8] O. B. Bayazit, G. Song, and N. M. Amato. Enhancing randomized motion planners: Exploring with haptic hints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 529–536, 2000.

[9] A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq, and V. S. Pande. Folding@home: Lessons from eight years of volunteer distributed computing. In

Proc. International Parallel and Distributed Processing Symposium (IPDPS), pages 1–8, Atlanta, Georgia, April 2009.

[10] A. Becker, C. Ertel, and J. McLurkin. Crowdsourcing swarm manipulation experiments: A massive online user study with large swarms of simple robots. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2825–2830, May 2014.

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

[12] A. A. Buss, Harshvardhan, I. Papadopoulos, O. Pearce, T. G. Smith, G. Tanase, N. Thomas, X. Xu, M. Bianco, N. M. Amato, and L. Rauchwerger. STAPL: standard template adaptive parallel library. In *Proc. of of SYSTOR 2010: The 3rd Annual Haifa Experimental Systems Conference, Haifa, Israel, May 24-26, 2010*, pages 1–10, New York, NY, USA, 2010. ACM.

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

[14] C. Chen, M. Rickert, and A. Knoll. Combining space exploration and heuristic search in online motion planning for nonholonomic vehicles. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 1307–1312, June 2013.

[15] C. Chen, M. Rickert, and A. Knoll. A traffic knowledge aided vehicle motion planning engine based on space exploration guided heuristic search. In *Intelligent Vehicles Symposium (V), 2014 IEEE*, pages 535–540, June 2014.

[16] C. Chen, M. Rickert, and A. Knoll. Kinodynamic motion planning with space-time exploration guided heuristic search for car-like robots in dynamic environ-

ments. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 2666–2671, Sept 2015.

[17] P. Cheng and S. LaValle. Reducing metric sensitivity in randomized trajectory design. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, volume 1, pages 43–48 vol.1, 2001.

[18] H. Choset and J. Burdick. Sensor-based exploration: The hierarchial generalized voronoi graph. *Int. J. Robot. Res.*, 19(2):96–125, 2000.

[19] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[20] K. D. Chrystopher L. Nehaniv. Like me?- measures of correspondence and imitation. *Cybernetics and Systems*, 32(1-2):11–51, 2001.

[21] H. Y. (Cindy), S. L. Thomas, D. Eppstein, and N. M. Amato. UOBPRM: A uniformly distributed obstacle-based PRM. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 2655–2662, 2012.

[22] J. Denny and N. M. Amato. Toggle PRM: simultaneous mapping of c-free and c-obstacle - a study in 2d -. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 2632–2639, 2011.

[23] J. Denny and N. M. Amato. The toggle local planner for sampling-based motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1779–1786, 2012.

[24] J. Denny and N. M. Amato. Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension. In *Algorithmic Foundations of Robotics*

X, volume 86 of *Springer Tracts in Advanced Robotics*, pages 297–312. Springer, Berlin/Heidelberg, 2013.

[25] J. Denny, J. Colbert, H. Qin, and N. M. Amato. On the theory of user-guided planning. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, Deajoon, Korea, October 2016.

[26] J. Denny, R. Sandstrom, and N. M. Amato. A general region-based framework for collaborative planning. In *The International Symposium on Robotics Research (ISRR)*, Genova, Italy, September 2015.

[27] J. Denny, R. Sandstrom, N. Julian, and N. M. Amato. A region-based strategy for collaborative roadmap construction. In *Algorithmic Foundations of Robotics XI*, pages 125–141, 2015.

[28] A. Dobson and K. E. Bekris. Sparse roadmap spanners for asymptotically near-optimal motion planning. *Int. J. Robot. Res.*, 2013.

[29] H. Doraiswamy and V. Natarajan. Efficient algorithms for computing reeb graphs. *Comput. Geom. Theory Appl.*, 42(6-7):606–616, Aug. 2009.

[30] C. Ekenna, S. A. Jacobs, S. Thomas, and N. M. Amato. Adaptive neighbor connection for PRMs: A natural fit for heterogeneous environments and parallelism. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1–8, Tokyo, Japan, November 2013.

[31] M. Foskey, M. Garber, M. C. Lin, and D. Manocha. A voronoi-based hybrid motion planner for rigid bodies. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 55–60, 2001.

[32] C. Guo, T. Tarn, N. Xi, and A. Bejczy. Fusion of human and machine intelligence for telerobotic systems. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages

3110–3115, 1995.

[33] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, SCC-4(2):100–107, 1968.

[34] A. Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, New York, 2001.

[35] K. Hauser. The minimum constraint removal problem with three robotics applications. *Int. J. Robot. Res.*, 33(1):5–17, 2014.

[36] K. Hauser and J.-C. Latombe. Multi-modal motion planning in non-expansive spaces. *Int. J. Robot. Res.*, 29(7):897–915, 2010.

[37] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proc. of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 203–212, New York, NY, USA, 2001. ACM.

[38] P. F. Hokayem and M. W. Spong. Bilateral teleoperation: An historical survey. In *Automatica 42*, pages 2035–2057, 2006.

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

[40] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Algorithmic Foundations of Robotics III*, pages 141–153, 1998.

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

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

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

[44] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3885–3891, 2005.

[45] Y. Hwang, K. Cho, S. Lee, S. Park, and S. Kang. Human computer cooperation in interactive motion planning. In *Proc. IEEE Int. Conf. Adv. Robot. (ICAR)*, pages 571–576, 1997.

[46] Y. K. Hwang and P. C. Chen. A heuristic and complete planner for the classical mover's problem. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 729–736, 1995.

[47] I. Ivanisevic and V. Lumelsky. Augmenting human performance in motion planning tasks- the configuration space approach. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2649–2654, 2001.

[48] I. Ivanisevic and V. J. Lumelsky. Configuration space as a means for augmenting human performance in teleoperation tasks. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 30(3):471–484, Jun 2000.

[49] L. Jaillet, J. Cortés, and T. Siméon. Sampling-based path planning on configuration-space costmaps. *Trans. Rob.*, 26(4):635–646, Aug. 2010.

[50] M. Kalisiak and M. van de Panne. RRT-blossom: RRT with a local flood-fill behavior. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1237–1242,

May 2006.

[51] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.*, 30:846–894, 2011.

[52] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe. Analysis of probabilistic roadmaps for path planning. In *IEEE Trans. Robot. Automat.*, volume 14, pages 166–171, 1998.

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

[54] P. Kormushev, S. Calinon, and D. G. Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 3232–3237, Oct 2010.

[55] J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 995–1001, 2000.

[56] KUKA Robotics Corporation. Kuka youbot store. `http://www.youbot-store.com`. Accessed: June 1, 2013.

[57] H. Kurniawati and D. Hsu. Workspace-based connectivity oracle - an adaptive sampling strategy for prm planning. In *Algorithmic Foundation of Robotics VII*, pages 35–51. Springer, Berlin/Heidelberg, 2008.

[58] A. Ladd and L. Kavraki. Measure theoretic analysis of probabilistic path planning. *IEEE Trans. Robot. Automat.*, 20(2):229–242, April 2004.

[59] N. Ladeveze, J.-Y. Fourquet, B. Puel, and M. Taix. Haptic assembly and disassembly task assistance using interactive path planning. In *Virtual Reality Conference, 2009. VR 2009. IEEE*, pages 19–25, March 2009.

[60] S. LaValle, J. Yakey, and L. Kavraki. A probabilistic roadmap approach for systems with closed kinematic chains. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1671–1676, Detroit, MI, 1999.

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

[62] J. Lee, O. Kwon, L. Zhang, and S. Yoon. Sr-rrt: Selective retraction-based rrt planner. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2543–2550, 2012.

[63] S. Lee, G. Sukhatme, G. J. Kim, and C.-M. Park. Haptic teleoperation of a mobile robot: A user study. *Presence: Teleoperators and Virtual Environments*, 14(3):345–365, 2005.

[64] J.-M. Lien, O. B. Bayazit, R.-T. Sowell, S. Rodriguez, and N. M. Amato. Shepherding behaviors. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4159–4164, April 2004.

[65] J.-M. Lien and E. Pratt. Interactive planning for shepherd motion. In *Proc. AAAI Spring Symposium*. AAAI, 2009.

[66] M. C. Lin. *Efficient Collision Detection for Animation and Robotics*. PhD thesis, University of California, Berkeley, CA, Dec. 1993.

[67] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[68] J. D. Marble and K. E. Bekris. Asymptotically near-optimal planning with probabilistic roadmap spanners. *IEEE Trans. Robot.*, 29:432–444, 2013.

[69] C. Masone, A. Franchi, H. H. Bulthoff, and P. R. Giordano. Interactive planning of persistent trajectories for human-assisted navigation of mobile robots. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 2641–2648, 2012.

[70] T. McMahon, S. Thomas, and N. M. Amato. Sampling based motion planning with reachable volumes: Application to manipulators and closed chain systems. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 3705–3712, Chicago, Il., Sept. 2014.

[71] T. McMahon, S. Thomas, and N. M. Amato. Sampling based motion planning with reachable volumes: Theoretical foundations. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 6514–6521, Hong Kong, China, May 2014.

[72] T. McMahon, S. L. Thomas, and N. M. Amato. Reachable volume RRT. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2977–2984, Seattle, Wa., May 2015.

[73] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato. A machine learning approach for feature-sensitive motion planning. In *Algorithmic Foundations of Robotics VI*, Springer Tracts in Advanced Robotics, pages 361–376. Springer, Berlin/Heidelberg, 2005.

[74] M. A. Morales A., R. Pearce, and N. M. Amato. Metrics for analyzing the evolution of C-Space models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1268–1273, May 2006.

[75] M. A. Morales A., L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato. C-space subdivision and integration in feature-sensitive motion planning. In *Proc. IEEE*

Int. Conf. Robot. Autom. (ICRA), pages 3114–3119, April 2005.

[76] J. Pan, L. Zhang, and D. Manocha. Retraction-based RRT planner for articulated models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2529–2536, 2010.

[77] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust online computation of reeb graphs: Simplicity and speed. *ACM Trans. Graph.*, 26(3):58.1–58.9, July 2007.

[78] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki. Sampling-based roadmap of trees for parallel motion planning. *IEEE Trans. Robot.*, 21(4):597–608, August 2005.

[79] E. Plaku, L. Kavraki, and M. Vardi. Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Trans. Robot.*, 26(3):469–482, June 2010.

[80] G. Reeb. Sur les points singuliers d'une forme de pfaff complement integrable ou d'une fonction numerique. *Comptes Rendus Acad. Sciences Paris*, 222:847–849, 1946.

[81] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, October 1979.

[82] S. Rodriguez and N. M. Amato. Utilizing roadmaps in evacuation planning. *Intern. J. Virtual Reality*, 10(1):67–73, 2011.

[83] S. Rodríguez, J. Denny, J. Burgos, A. Mahadevan, K. Manavi, L. Murray, A. Kodochygov, T. Zourntos, and N. M. Amato. Toward realistic pursuit-

evasion using a roadmap-based approach. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1738–1745, 2011.

[84] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato. An obstacle-based rapidly-exploring random tree. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 895–900, 2006.

[85] S. Rodriguez, S. Thomas, R. Pearce, and N. M. Amato. (RESAMPL): A region-sensitive adaptive motion planner. In *Algorithmic Foundation of Robotics VII*, pages 285–300. Springer, Berlin/Heidelberg, 2008.

[86] K. Shi, J. Denny, and N. M. Amato. Spark PRM: Using RRTs within PRMs to efficiently explore narrow passages. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, P. R. China, June 2014.

[87] A. Shkolnik, M. Walter, and R. Tedrake. Reachability-guided sampling for planning under differential constraints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2859–2865, May 2009.

[88] H. Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 41(2):11:1–11:36, Feb. 2015.

[89] T. Simeon, J.-P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6):477–493, 2000.

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

[91] G. Song and N. M. Amato. A motion planning approach to folding: From paper craft to protein folding. *IEEE Trans. Robot. Automat.*, 20:60–71, February 2004.

[92] G. Song, S. Thomas, K. Dill, J. Scholtz, and N. Amato. A path planning-based study of protein folding with a case study of hairpin formation in protein G and L. In *Proc. Pacific Symposium of Biocomputing (PSB)*, pages 240–251, 2003.

[93] M. Strandberg. Augmenting RRT-planners with local trees. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 4, pages 3258–3262, 2004.

[94] M. Taïx, D. Flavigné, and E. Ferré. Human interaction with motion planning algorithm. *Journal of Intelligent & Robotic Systems*, 67(3-4):285–306, 2012.

[95] X. Tang, B. Kirkpatrick, S. Thomas, G. Song, and N. M. Amato. Using motion planning to study RNA folding kinetics. *J. Comput. Biol.*, 12(6):862–881, 2005.

[96] X. Tang, S. L. Thomas, P. Coleman, and N. M. Amato. Reachable distance space: Efficient sampling-based planning for spatially constrained systems. *Int. J. Robot. Res.*, 29(7):916–934, 2010.

[97] L. Tapia, S. L. Thomas, B. Boyd, and N. M. Amato. An unsupervised adaptive strategy for constructing probabilistic roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4037–4044, 2009.

[98] J. P. van den Berg and M. H. Overmars. Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. *Int. J. Robot. Res.*, 24(12):1055–1071, 2005.

[99] A. Vargas Estrada, J.-M. Lien, and N. M. Amato. Vizmo++: a visualization, authoring, and educational tool for motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 727–732, May 2006.

[100] Z. Wood, H. Hoppe, M. Desbrun, and P. Schröder. Removing excess topology from isosurfaces. *ACM Trans. Graph.*, 23(2):190–208, Apr. 2004.

[101] Z. J. Wood, P. Schröder, D. Breen, and M. Desbrun. Semi-regular mesh extraction from volumes. In *Proc. of the Conference on Visualization '00*, VIS '00, pages 275–282, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[102] Y. Yan, E. Poirson, and F. Bennis. Integrating user to minimize assembly path planning time in plm. In *Product Lifecycle Management for Society*, volume 409 of *IFIP Advances in Information and Communication Technology*, pages 471–480. Springer Berlin Heidelberg, 2013.

[103] A. Yershova, L. Jaillet, T. Simeon, and S. M. Lavalle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3856–3861, April 2005.

[104] L. Zhang and D. Manocha. An efficient retraction-based RRT planner. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3743–3750, 2008.