MOUNT HOLYOKE COLLEGE

SENIOR THESIS

GIGPRM: A Gradually Improving Guided Probabilistic Roadmap Strategy

by

Anna Dai

Supervised by

Prof. Diane Uwacu

Presented to the faculty of Mount Holyoke College in partial fulfillment of the requirements for the degree of Bachelor of Arts in Computer Science

May 2026

Abstract

Skeleton-guided sampling-based motion planners, such as Dynamic Region PRM (DR-PRM), use a workspace skeleton to bias sampling toward topologically important regions of the environment, such as narrow passages. However, these methods assume that the skeleton is medially centered in the free space and that every skeleton vertex has sufficient clearance, meaning it is far enough from surrounding obstacles to allow valid configurations to be sampled in its vicinity. When this assumption is violated, structurally important regions may be discarded, and the planner may fail to find a feasible path.

This thesis introduces Gradually Improving Guided PRM (GIGPRM), a strategy designed to remain reliable when workspace skeletons are imperfect. GIGPRM preserves low-clearance skeleton vertices and incrementally relocates expansion regions toward feasible nearby space during roadmap construction. This allows skeleton guidance to be adjusted during planning.

We evaluate GIGPRM against DR-PRM in three benchmark environments under skeletons of varying quality, and show that GIGPRM achieves substantially higher path-finding rates, lower runtime, and dramatically fewer collision detection calls. Overall, GIG-PRM achieves the efficiency of guided sampling without being constrained by skeleton quality.

Acknowledgments

I would like to express my deepest gratitude to my thesis advisor, Professor Diane Uwacu, for her invaluable guidance, patience, and support throughout every stage of this work. Her insight and encouragement shaped the direction of this thesis and made the entire process a genuinely rewarding experience. Beyond this project, I am especially grateful to her for introducing me to the field of robotics and motion planning, and for inspiring the curiosity that led me here in the first place.

I am also thankful to my academic advisor, Professor Audrey St. John, and to the Mount Holyoke College Computer Science Department, for the mentorship, welcoming academic environment, and opportunities to grow as both a student and a researcher throughout my time at Mount Holyoke.

Finally, I am profoundly thankful to my family and friends for their constant love, encouragement, and belief in me. This work would not have been possible without them.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Sampling-based motion planning algorithms such as Probabilistic Roadmaps[12] and Rapidly-Exploring Random Trees[14] have become fundamental tools for solving high-dimensional planning problems. Their probabilistic completeness and scalability make them well suited for complex robotic systems. However, these methods rely heavily on uniform random sampling, which makes them particularly vulnerable to the narrow passage problem. When feasible paths lie in low-clearance regions that occupy only a small fraction of the configuration space, purely random exploration becomes inefficient and unreliable.

Skeleton-guided planners were introduced to address this limitation by incorporating workspace structure into the sampling process. Methods such as Dynamic Region PRM (DR-PRM)[18] leverage a workspace skeleton to bias sampling toward topologically meaningful passages. By guiding expansion along skeleton edges, these planners significantly improve performance. However, their effectiveness strongly depends on the quality and placement of the skeleton. If the skeleton is not well-centered in free space or has low clearance, expansion regions may fail to initialize or may become trapped near obstacles. As a result, planning may stall or slow significantly.

1.2 Research Questions

Existing dynamic-region methods assume that skeleton vertices have sufficient clearance for immediate sampling and expansion. When this assumption is violated, structurally important regions may be discarded during initialization, or expansion may fail during iteration. This project aims to address the following research questions:

- How can skeleton-guided planning approaches be adapted to remain reliable and efficient in environments where workspace skeletons may be imperfect?
- Can sampling in configuration space (C-space) be used to adjust workspace skeletons so they better reflect true connectivity and clearance?

To address this issue, we propose Gradually Improving Guided PRM (GIGPRM), an extension of DR-PRM that relaxes this rigid dependence on ideal skeleton placement. Instead of discarding low-clearance skeleton vertices, GIGPRM preserves them and introduces a dynamic relocation mechanism during iteration. When no valid expansion region can be selected, the planner incrementally shifts region centers toward feasible nearby space based on clearance evaluation. This allows the planner to gradually refine skeleton guidance during roadmap construction rather than requiring it to be perfect from the start. The central idea is to treat the skeleton as an adjustable guidance structure rather than a fixed constraint.

1.3 Thesis Structure

The rest of this thesis is organized as follows. Chapter 2 reviews related works including sampling-based motion planning and skeleton-guided methods. Chapter 3 presents the proposed GIGPRM algorithm. Chapter 4 describes the experimental setup, reports the performance of GIGPRM and Guided-PRM across three benchmark environments under skeletons of varying quality, and discusses the observed results. Chapter 5 concludes the thesis and outlines directions for future work.

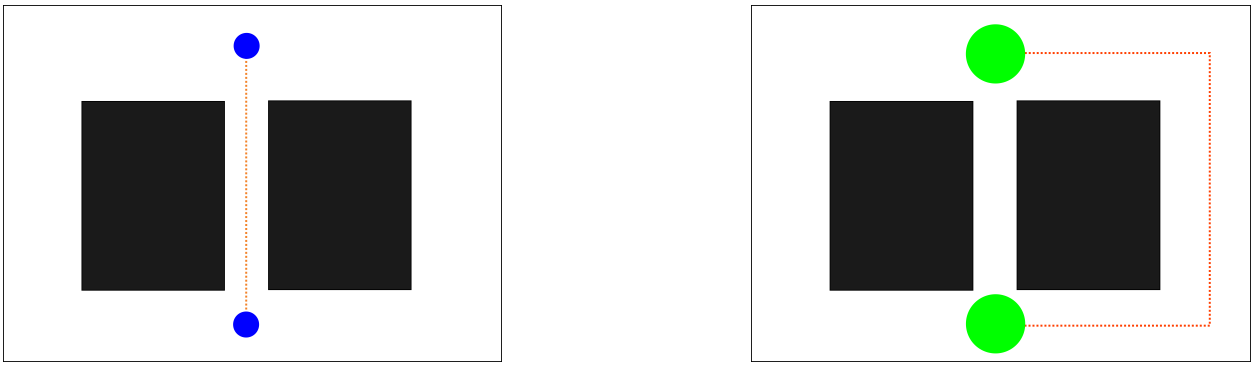
Chapter 2

Literature Review

2.1 Motion Planning

Motion planning is a fundamental problem in robotics, which is about computing feasible motions that allow a system to move from a start configuration to a goal configuration while avoiding obstacles and satisfying physical constraints. At a high level, this problem can be divided into two distinct subproblems. The first is path planning, which finds a collision-free path through the environment without considering how the robot physically moves along it. The second is motion planning, which accounts for the robot's physical properties such as its shape and dynamic constraints, determining not just where the robot needs to go but how it can feasibly get there. As illustrated in Figure 2.1, a small robot can follow a direct path straight through the passage between two obstacles. A larger robot, however, cannot fit through the same passage and must instead plan a longer path around the obstacles. This explains the key distinction: path planning finds a geometric route through the environment, while motion planning additionally accounts for the physical properties of the robot to determine how it can navigate.

A central concept in motion planning is the configuration space, which represents all possible states of the robot as points in a higher-dimensional space, allowing the planning



(a) (b)

Figure 2.1: Both robots share the same planned path through the narrow passage. (a) A small robot can successfully follow the path, as its size allows it to fit through the passage. (b) A large robot cannot follow the same path because its physical size makes it infeasible, and motion planning must find an alternative route around the obstacles.

problem to be reformulated as the search for a collision-free path within this space[13]. By transforming both the robot and its environment into this representation, complex interactions between the robot's geometry and obstacles can be systematically analyzed. Despite this abstraction, motion planning remains challenging due to the high dimensionality of the configuration space and the complexity of real-world environments. As a result, a wide range of algorithmic approaches have been developed to address these challenges.

2.2 Sampling-Based Planners

Sampling-based motion planning algorithms address motion planning in high-dimensional spaces by randomly sampling the configuration space and constructing connectivity between feasible configurations[6]. They have become a central tool for solving high-dimensional planning problems that are intractable for classical grid-based or exact methods. Among these algorithms, Probabilistic Roadmaps (PRM) and Rapidly-Exploring Random Trees (RRT) are two foundational approaches that have demonstrated strong performance across a wide range of robotic systems and environments.

2.2.1 Probabilistic Roadmaps

The Probabilistic Roadmaps Method (PRM)[12] is a multi-query planner that constructs a graph-based approximation of the robot's configuration space. During an initial learning phase, PRM samples configurations uniformly at random and retains those that are collision-free as nodes in a roadmap. Nearby nodes are then connected using a local planner, provided that the resulting paths do not intersect obstacles. Over time, this process produces a graph that captures the global connectivity of the free space, as illustrated in Figure 2.2. In the query phase, planning problems are solved by connecting the start and goal configurations to the roadmap and searching for a path within the graph. This separation between roadmap construction and query resolution makes PRM particularly effective in static environments where multiple planning queries are expected.

PRM performs well in high-dimensional spaces and environments with large open regions, where random sampling can efficiently approximate connectivity. However, PRM faces two fundamental challenges. First, it requires solving a boundary value problem when attempting to connect two configurations, that is, determining whether a collision-free motion exists between a given pair of states while satisfying the system's motion constraints. This task can be difficult or computationally expensive for systems with complex or nonholonomic dynamics, where the robot is subject to motion constraints that restrict the directions it can move instantaneously, such as a car that can move forward and backward but cannot slide sideways. This limits PRM's applicability primarily to kinematic planning problems. Second, PRM suffers from the narrow passage problem[21], where regions of low clearance are unlikely to be sampled using uniform random sampling. Although this issue can be alleviated in low-dimensional spaces by biasing samples near obstacles or along the medial axis, it remains a significant limitation in general settings.

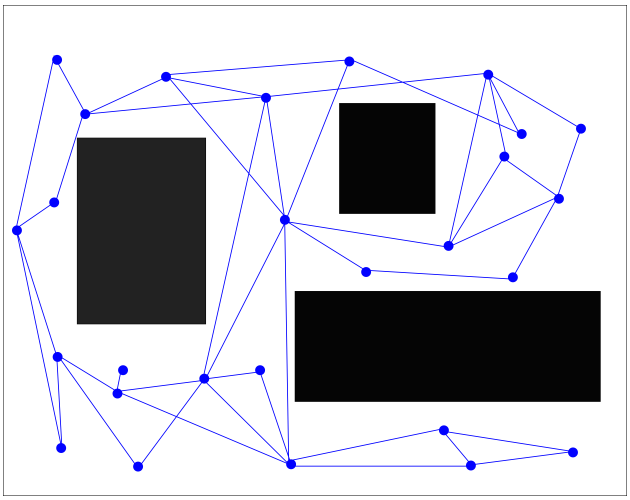


Figure 2.2: Illustration of a Probabilistic Roadmap (PRM). Random samples are connected to form a graph that captures the connectivity of the free space.

2.2.2 Rapidly-Exploring Random Trees

Rapidly-Exploring Random Trees (RRT)[14] were introduced to address several of these limitations, particularly the reliance on solving boundary value problems in roadmap-based planners. Unlike PRM, RRT is a single-query, incremental planner that grows a tree rooted at the initial configuration by extending feasible motions forward, rather than attempting to directly connect arbitrary pairs of configurations.

At each iteration, the algorithm samples a random state and extends the tree toward it by selecting the nearest existing node and generating a short feasible motion. This mechanism induces a strong bias toward unexplored regions of the state space, allowing RRT to rapidly expand into new areas without explicitly maintaining a global roadmap, as shown in Figure 2.3. As a result, RRT is well suited for online planning problems, high-dimensional spaces, and systems with nonholonomic or kinodynamic constraints.

Despite their differences in structure and intended use cases, PRM and RRT share important theoretical properties. Both algorithms are probabilistically complete, meaning that if a feasible solution exists, the probability that the algorithm fails to find one approaches zero as the number of samples increases. However, probabilistic completeness does not imply

guarantees on solution quality. In practice, both PRM and RRT are primarily designed to find feasible paths quickly, rather than to minimize path cost. Moreover, like PRM, RRT also suffers from the narrow passage problem, as its reliance on uniform random sampling makes it unlikely to efficiently explore low-clearance regions of the configuration space. This limitation has been observed in practice, where tree-based planners such as RRT struggle to efficiently explore constrained regions such as narrow passages[17].

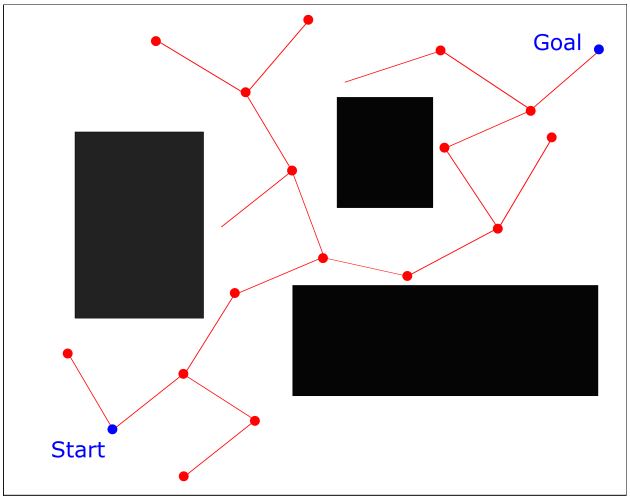


Figure 2.3: Illustration of a Rapidly-Exploring Random Tree (RRT). A tree is incrementally grown from the start configuration toward unexplored regions until the goal is reached.

2.2.3 Variants and Improvements

Karaman and Frazzoli formally analyzed this limitation and showed that standard PRM and RRT are not asymptotically optimal[11]. Specifically, they proved that under mild technical conditions, the cost of the solution returned by these algorithms converges almost surely to a value that is strictly worse than the optimal solution as the number of samples increases. In PRM, this behavior arises from fixed connection strategies that prevent the roadmap from refining toward the optimal path. In RRT, the tree structure locks in early decisions, and once a node is added, its parent cannot be changed, even if better connections become available later. As a result, increasing the number of samples does not systematically improve path quality.

To address this issue, Karaman and Frazzoli introduced PRM* and RRT*, which modify the original algorithms to guarantee asymptotic optimality while preserving probabilistic completeness and computational efficiency. PRM* extends PRM by adapting the connection strategy as the number of samples increases. Instead of using a fixed radius or a fixed number of neighbors, PRM* adapts its connection radius as the roadmap grows. The radius gradually decreases with the number of samples, while accounting for the dimensionality of the configuration space. This scaling ensures that the roadmap becomes sufficiently dense to approximate the optimal path while avoiding unnecessary connections. Under this scheme, the shortest path in the PRM* roadmap converges almost surely to the optimal solution.

RRT* similarly improves RRT by introducing cost-aware tree growth and a rewiring mechanism. When a new node is added, RRT* selects as its parent the nearby node that minimizes the cost from the root, rather than simply choosing the nearest neighbor. The algorithm then attempts to rewire existing nearby nodes through the new node if doing so reduces their path cost. This rewiring step allows the tree to continuously refine its structure, correcting suboptimal decisions made earlier in the planning process. As more samples are added, the solution produced by RRT* converges almost surely to the optimal path, while maintaining time and memory complexity comparable to that of standard RRT. While PRM* and RRT* successfully address the lack of optimality in classical sampling-based planners, their theoretical guarantees rely on a key assumption: the existence of an optimal path with positive clearance from obstacles. This assumption excludes environments containing extremely narrow passages, where valid paths lie arbitrarily close to obstacles and occupy regions of very small measure. In such cases, uniform random sampling is unlikely to generate sufficient samples within the passage, limiting the planner's ability to discover feasible or optimal solutions. Consequently, although PRM* and RRT* improve path quality once relevant regions have been sampled, they do not fundamentally resolve the difficulty of exploring narrow passages, motivating further research into structure-aware and

clearance-sensitive planning methods. More recent work has also explored learning-based extensions, in which sampling distributions are learned from prior planning experience to bias exploration toward promising regions[8]. While effective, such methods require representative training data, whereas the skeleton-guided approaches considered in this thesis derive guidance directly from the workspace geometry.

2.3 The Narrow Passage Problem

The narrow passage problem[21] refers to the difficulty sampling-based motion planners face when valid paths must pass through regions of the configuration space that have very small clearance. This challenge arises because sampling-based planners rely on random sampling to approximate the connectivity of the free configuration space. Although these regions are critical for feasible motion, they occupy only a tiny fraction of the configuration space volume. As a result, uniform random sampling is unlikely to generate sufficient samples inside narrow passages, especially as dimensionality increases, as illustrated in Figure 2.4. Hsu, Kavraki, Latombe, Motwani, and Sorkin[7] formally characterized this difficulty through properties such as ε-goodness, expansiveness, and path clearance, showing that the number of samples required to capture connectivity grows sharply when narrow passages are present.

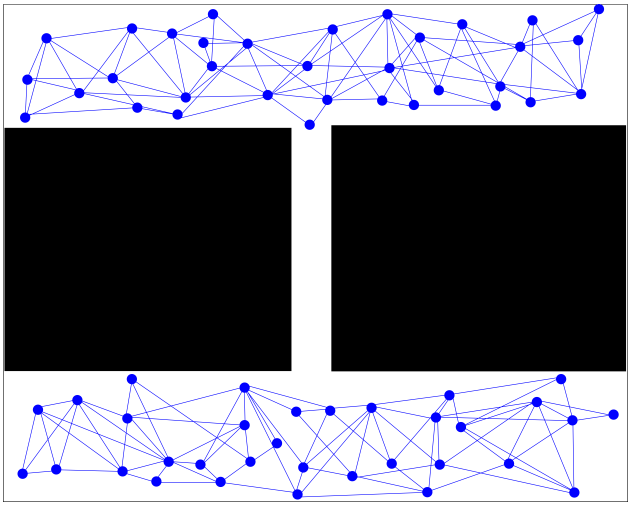


Figure 2.4: Illustration of the narrow passage problem. Although feasible paths exist through the free space, narrow regions between obstacles are difficult to sample due to their small volume relative to the overall configuration space. As a result, uniform random sampling produces sparse coverage in these critical regions.

2.3.1 Gaussian Sampler

Gaussian sampling[2] is an early obstacle-based biased sampling strategy proposed to improve the performance of probabilistic roadmap planners in environments containing narrow passages. Instead of drawing samples uniformly from the configuration space, Gaussian sampling generates pairs of samples from a Gaussian distribution centered at a randomly chosen configuration. If one sample lies in collision and the other is collision-free, the collision-free configuration is retained. This mechanism increases the likelihood of accepting samples near obstacle boundaries, under the assumption that narrow passages tend to occur in proximity to obstacles, as illustrated in Figure 2.5.

Gaussian sampling is effective for the narrow passage problem because narrow passages are typically located near obstacles, where small perturbations can cause collisions. By preferentially accepting configurations that are close to colliding configurations, Gaussian sampling increases the probability of placing samples in regions of low clearance that are unlikely to be discovered through uniform random sampling. In this sense, the method implicitly "blurs" obstacles in the configuration space, increasing sample density in difficult

regions while reducing oversampling in large, open areas.

One of the primary advantages of Gaussian sampling is its simplicity and generality. The method relies only on collision checking and does not require computation of global geometric structures such as medial axes or visibility regions, making it applicable to a wide range of planning problems. Gaussian sampling has been shown to significantly reduce the number of roadmap nodes required to solve problems dominated by narrow passages, thereby decreasing the overall cost of local planning and graph construction. In environments with high obstacle density or tight corridors, the approach can outperform uniform sampling by large margins, both in terms of success rate and computation time. These properties have motivated extensions of Gaussian sampling to other sampling-based frameworks, including RRT[9].

However, it also has several limitations. It introduces additional computational overhead because many sampled configuration pairs are rejected after collision checking. Its effectiveness is also highly sensitive to the choice of the Gaussian variance parameter, which requires careful tuning and is problem-dependent. Moreover, the method is most beneficial in environments dominated by narrow passages and may be less efficient in largely open spaces, while providing no guarantees on solution optimality.

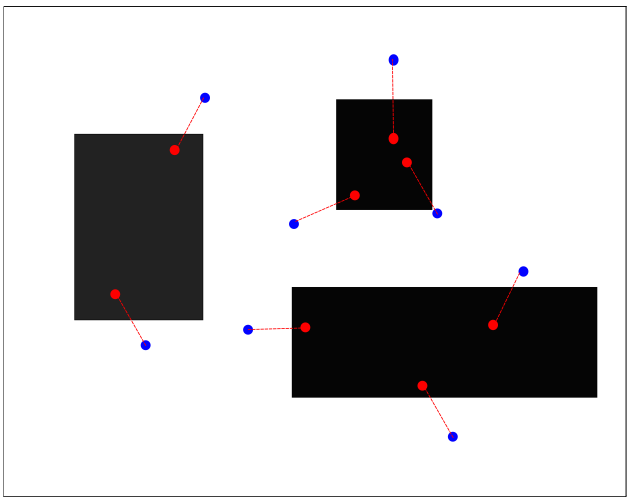


Figure 2.5: Illustration of Gaussian sampling. Pairs of nearby configurations are generated, where one lies in collision (red) and the other is collision-free (blue). Collision-free samples that are close to obstacles are retained, biasing sampling toward obstacle boundaries.

2.3.2 Bridge Test

The bridge test[20] is a sampling strategy explicitly designed to detect narrow passages using only local collision information. In the bridge test, two nearby configurations are sampled such that both lie in collision, while the midpoint between them is collision-free. This midpoint is then accepted as a valid sample, as illustrated in Figure 2.6. This geometric condition is significantly more likely to occur inside narrow passages than in wide-open regions, making the bridge test an effective filter for identifying configurations in low-clearance areas. It does not require global geometric computation and is applicable in higher-dimensional spaces.

However, there are also limitations: the test may generate false positives near obstacle corners, and its higher per-sample collision-checking cost can reduce efficiency if used in isolation. To balance these trade-offs, the bridge test is often combined with uniform sampling in a hybrid strategy, where uniform samples ensure global coverage while bridge-test samples improve connectivity through narrow passages. While this approach substantially improves feasibility in difficult environments, it remains heuristic in nature and does not provide guarantees of completeness or optimality on its own. More recent work has extended the bridge test with directional perturbations and filtering to suppress false positives near obstacle boundaries, and uses clustered bridge points as a sparse structural prior for global guidance[4]. However, the framework introduces several hyperparameters that require further tuning, and the validation cost increases with the dimensionality of the configuration space.

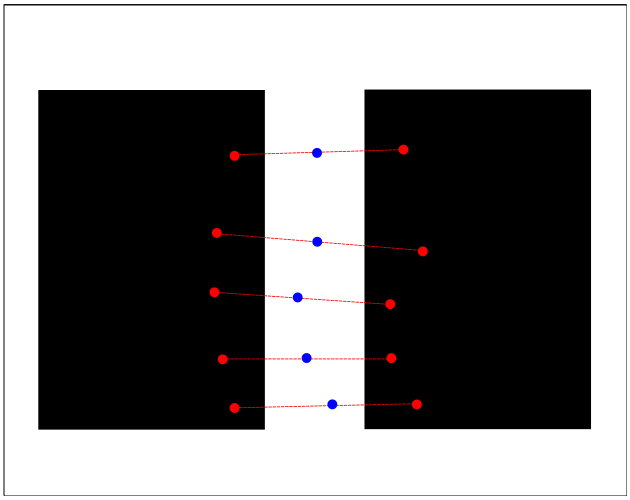


Figure 2.6: Illustration of the bridge test. Two nearby configurations in collision (red) are sampled, and their midpoint (blue) is collision-free. Such midpoints are likely to lie within narrow passages and are therefore retained.

2.3.3 Visibility-based PRM

Visibility-based Probabilistic Roadmaps (VisPRM)[19] were proposed as an extension of the classical PRM framework to address challenges arising from uniform random sampling, particularly in environments containing narrow passages. It seeks to construct compact roadmaps that explicitly encode connectivity while reducing unnecessary sampling and connection attempts.

The central concept underlying VisPRM is visibility in the configuration space, defined with respect to a local planner. Rather than adding every collision-free sample to the roadmap, VisPRM selectively retains only those samples that expand coverage or improve connectivity. The algorithm classifies accepted samples into two categories: guard nodes and connection nodes. A guard node represents a new region of the configuration space whose visibility domain is not covered by any existing guard. If a sampled configuration cannot be connected to any existing guard using the local planner, it is added as a new guard. In contrast, a connection node is introduced when a sampled configuration can connect to guards belonging to different connected components of the roadmap, thereby serving as a

bridge between them. Samples that only connect to a single existing guard are discarded, as they do not improve coverage or connectivity. By iteratively adding only guards and connection nodes, VisPRM constructs a sparse roadmap that captures the essential connectivity of the free space. The algorithm terminates based on a probabilistic stopping criterion that estimates when sufficient coverage has been achieved. Subsequent work has explored enhancing sampling strategies based on visibility, aiming to generate configurations with higher clearance and improved connectivity properties. For example, Kala[10] proposes sampling techniques that explicitly maximize sample visibility by positioning samples near the centers of narrow corridors, which improves edge construction and accelerates solution discovery.

A key advantage of VisPRM in addressing the narrow passage problem lies in its visibility-based sampling strategy. Instead of relying on dense sampling, VisPRM identifies configurations that connect guards belonging to different components and retains them as connection nodes. Since narrow passages often serve as connectors between larger free regions, this mechanism allows VisPRM to detect and represent such passages efficiently without extensive sampling inside them. As a result, narrow passages can be captured with relatively few samples while still preserving global connectivity.

However, the planner still relies on random sampling and may reject many samples during visibility checks, which can increase computational overhead. Its performance also depends on the choice of local planner used to define visibility. Moreover, while VisPRM improves connectivity in narrow passages, it does not provide guarantees on path optimality or solution quality.

2.4 Skeleton-Guided Planners

Workspace-guided planners are a class of motion-planning methods that use workspace information to inform the sampling process. Skeleton-guided planners are a prominent type of workspace-guided planner that use a workspace skeleton, which is a representation that

captures the connectivity of the free space at a higher level. The skeleton is not intended to be a complete roadmap, but instead serves as a guidance structure that biases sampling toward regions that are likely to contain feasible paths, particularly in environments with narrow passages or complex topology.

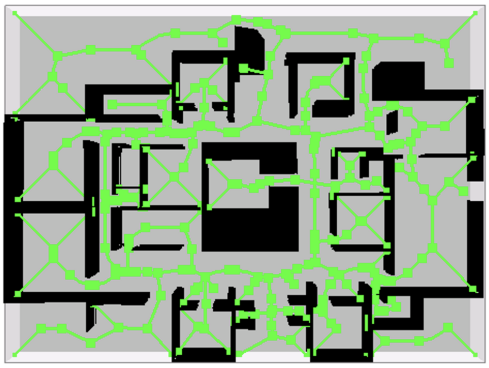
Skeleton-guided planners include methods such as Dynamic Region-biased Rapidly-Exploring Random Trees (DR-RRT)[3], Dynamic Region PRM (DR-PRM)[18], Hierarchical Annotated Skeleton Planning (HASP)[23], and Hierarchical Annotated-Skeleton Guided RRT (HAS-RRT)[24]. DR-RRT and DR-PRM both work by moving sampling regions along the workspace skeleton to focus exploration through structurally significant corridors. HASP and HAS-RRT build on this by introducing a hierarchical search strategy that prioritizes paths indicated by the skeleton first and falls back to local exploration only when needed. Those strategies represent the current state of the art within this category and have been shown to outperform general sampling heuristics like the ones mentioned in previous sections, and other workspace-based planners that doesn't use skeleton graphs such as Synergistic Combination of Layers of Planning (SyCLoP)[15] and the Exploration-Exploitation tree (EET) strategy[16], achieving faster roadmap construction and higher reliability in structured environments.

2.4.1 Workspace Skeleton

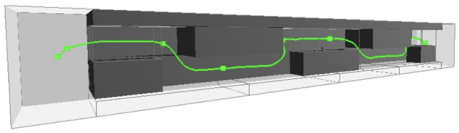
A workspace skeleton is a compact, low-dimensional graph embedded in the free workspace that captures its topological connectivity. Formally, it is defined as a deformation retract of the free workspace, meaning that every point in the free space can be mapped onto the skeleton while preserving the topology of the space[3]. As a result, the skeleton serves as a minimal representation of the free workspace that a planner can follow to guide exploration. However, it is important to note that the skeleton encodes only the geometric path through the environment and does not account for the robot's physical properties such as its size or shape. Therefore, skeleton guidance addresses the path planning component of the problem

but does not by itself solve the full motion planning problem.

There are several methods for computing workspace skeletons depending on the dimensionality of the environment. In two-dimensional environments, the medial axis skeleton[1] is commonly used, defined as the locus of centers of all maximal inscribed circles within the free space. In three-dimensional environments, the mean curvature skeleton [22] can be used to capture the connectivity of more complex space by iteratively contracting the boundary of the free space inward until it collapses into a compact curve that reflects the shape of the environment. Figure 2.7 shows examples of workspace skeletons computed for two-dimensional and three-dimensional environments using the medial axis and mean curvature methods respectively.



(a) Medial axis skeleton of a 2D building floor plan.



(b) Mean curvature skeleton of a 3D environment.

Figure 2.7: Examples of workspace skeletons (shown in green) computed for 2D and 3D environments. (a) The medial axis skeleton of a 2D building floor plan traces the center of each corridor and room, capturing the navigable connectivity of the environment. (b) The mean curvature skeleton of a 3D environment is computed by progressively contracting the boundary of the free space inward until it collapses into a compact curve through the navigable space. Figure (b) is reproduced from [24].

In practice, skeletons are constructed as a preprocessing step using computational

geometry libraries such as the Computational Geometry Algorithms Library (CGAL) [5]. However, the algorithms provided by such libraries produce approximate skeletons that are not always accurate, and the quality of the resulting skeleton degrades as the environment becomes more complex. Figure 2.8 illustrates this in the Mining environment, where the planner must navigate through narrow tunnels embedded inside solid structures, meaning the skeleton must lie within the tunnels to be useful for planning. As shown in Figure 2.8(a), the default skeleton produced automatically by CGAL is severely misplaced, lying almost entirely outside the tunnels and failing to capture the navigable structure of the environment. Producing a usable skeleton therefore requires manual pruning and curation, as shown in Figure 2.8(b), which was built by graduate students in the Parasol Lab, requiring considerable time and effort.

(a) (b)

Figure 2.8: Comparison of default and curated skeletons (shown in green) for the Mining environment. (a) The default skeleton generated automatically by CGAL. (b) The curated skeleton which provides a much more accurate representation of the environment's connectivity.

2.4.2 Dynamic Region-biased Rapidly-exploring Random Trees

Dynamic Region-biased Rapidly-Exploring Random Trees (DR-RRT)[3] guides the exploration of the basic RRT algorithm by explicitly guiding exploration using a workspace skeleton that encodes topological information about the environment. Standard RRT rapidly explores large open regions but struggles in maze-like environments where feasible paths lie within narrow corridors. DR-RRT addresses this limitation by directing sampling toward workspace regions that are topologically significant.

The approach begins with a preprocessing step in which a workspace embedding graph is constructed to represent the connectivity of the free workspace. This graph is typically implemented as a Reeb graph, which captures changes in the connectivity of workspace cross-sections and encodes the distinct topological ways a robot can navigate from one region to another. Given a planning query, a directed query skeleton is derived from this embedding graph to indicate promising exploration directions from the start toward unexplored regions of the workspace.

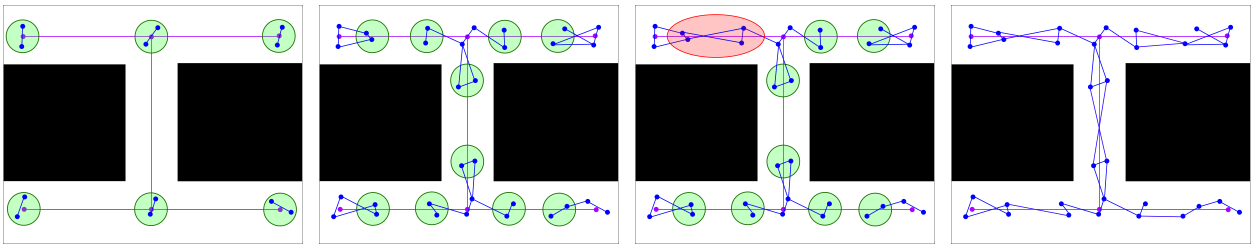
During planning, DR-RRT introduces dynamic sampling regions that move along the query skeleton as the RRT grows. Instead of sampling uniformly over the entire configuration space, random samples are generated preferentially within these regions, biasing tree expansion toward narrow corridors and unexplored passages suggested by the skeleton. As the tree reaches new portions of the skeleton, regions are updated, merged, or removed, allowing the planner to adaptively focus computational effort where it is most needed. Importantly, the planner retains a degree of uniform sampling to preserve probabilistic completeness. Experimental results demonstrate that DR-RRT significantly improves planning efficiency in environments characterized by narrow passages and complex topology.

Compared to standard RRT and other adaptive variants, DR-RRT achieves lower planning times and higher success rates, particularly in large maze-like workspaces where uniform sampling is ineffective. By combining global workspace topology with local sampling-based exploration, DR-RRT exemplifies how structure-aware guidance can overcome fundamental limitations of random sampling.

2.4.3 Dynamic Region PRM

Dynamic Region PRM (DR-PRM)[18] extends the idea of dynamic region sampling to the roadmap setting of PRM. Instead of sampling uniformly in configuration space, DR-PRM uses a workspace skeleton to guide its sampling. It initializes local connected components near skeleton vertices and assigns dynamic sampling regions that move along skeleton edges

as components expand. When components meet from opposite directions, they merge to form bridges, improving connectivity across the roadmap. Figure 2.9 illustrates this progression.



(a) Initialization (b) Expansion (c) Merging (d) Connected roadmap

Figure 2.9: Illustration of the DR-PRM algorithm. The workspace skeleton is shown in pink and the roadmap in blue. Green circles indicate active sampling regions and the red circle highlights a locally connected component. (a) Sampling regions are initialized at each skeleton vertex. (b) Regions expand along skeleton edges, growing local components. (c) A connection attempt is made between two local components. (d) The components successfully merge to form a bridge across the skeleton edge.

The implementation of DR-PRM is divided into two parts, including initialization and iteration, as illustrated in Algorithm 1. During initialization, the planner first constructs or loads a workspace skeleton. For each skeleton vertex v, the algorithm computes a region radius and samples collision-free configurations within the region centered at that vertex (Algorithm 1, Lines 3–5). These configurations are locally connected to form edges among nearby samples (Lines 7–10), producing a set of initial connected components. For each connected component, expansion regions are then initialized along the outgoing skeleton edges (Lines 11-13). The sampled configurations and edges are subsequently added to the roadmap, establishing the initial graph structure (Lines 14-15).

After initialization, the planner enters the main PRM loop. In each iteration, an expansion region is probabilistically selected (Line 19). If a valid region is available, the algorithm generates additional samples within that region to expand the corresponding local component (Line 21). As long as the sampled configurations remain inside the region, the region is advanced along the associated skeleton edge to continue exploration (Lines 22-23). The algorithm then attempts to connect nearby components along that edge to improve roadmap connectivity (Line 25). If no expansion region can be selected—for example, when

all regions have insufficient clearance—the algorithm instead selects a random unconnected skeleton segment and attempts to connect components directly across the environment (Lines 26-28).

Algorithm 2 defines how dynamic regions are initialized and advanced during roadmap construction. Regions are first created from local components at skeleton vertices (Lines 2–6). As a region moves along a skeleton edge, it either advances by updating its center and radius or, upon reaching the edge endpoint, attempts to merge nearby components through local connections (Lines 10–17). If no merge occurs, new regions are initialized at the target vertex to continue exploration, and the current region is removed. This process enables directed exploration along the skeleton while improving connectivity in the roadmap.

Compared to standard PRM, which relies on uniform random sampling and nearest-neighbor connections to gradually approximate free-space connectivity, DR-PRM actively guides exploration along structural corridors. Standard PRM may oversample open regions while still missing narrow passages, whereas DR-PRM focuses sampling along skeleton-defined segments to promote faster and more reliable connectivity in structured environments. However, its effectiveness relies on key assumptions: the robot is typically modeled as a rigid body of fixed size, and the skeleton is assumed to be centered within the free space. When these assumptions are violated, for example, if the skeleton has low clearance or lies far from the start and goal configurations, sampling may be misdirected, leading to reduced efficiency and degraded performance.

Algorithm 1 Roadmap construction with Dynamic Region PRM

Require: Skeleton $S = (S_V, S_E)$, Roadmap $G = (G_V, G_E)$

function BUILDROADMAP
  Initialize components at each skeleton vertex
  for all $v \in S_V$ **do**
    $r \leftarrow$ GETREGIONRADIUS(v.point)
    $Q \leftarrow$ SAMPLEVALIDCONFIGURATIONS($\beta_r(v)$)
    $E \leftarrow \emptyset$
    for all $q \in Q$ **do**
      $N \leftarrow$ NEARESTNEIGHBORS(q, Q)
      $E \leftarrow E \cup$ ATTEMPTCONNECTIONS(q, N)
    end for
    for all Connected Component $cc \in (Q, E)$ **do**
      INITIALIZEREGIONS(cc.vertices, v)
    end for
    $G_V \leftarrow G_V \cup Q$
    $G_E \leftarrow G_E \cup E$
  end for
  PRM Loop
  while $\neg done$ ▷ either node limit or S covered **do**
    $r \leftarrow$ SELECTREGION()
    if $r \neq \emptyset$ **then**
      $Q \leftarrow$ EXPANDLOCALCOMPONENT(r)
      while $\exists q \in Q \mid r$.CONTAINS(q) **do**
        ADVANCEREGION(r, Q)
      end while
      CONNECTLOCALCOMPONENTS(r.edge)
    else
      $e \leftarrow$ RANDOMUNCONNECTEDSEGMENT()
      CONNECTLOCALCOMPONENTS(e)
    end if
  end while
end function

Algorithm 2 Dynamic Region operations

```
***Initialize regions and local components***
function InitializeRegions(Configurations Q, SkeletonVertex v)
    for all e ∈ v.GetOutboundEdges() do
        C ← MakeLocalComponent(e, v, Q)
        CreateRegion(C)
    end for
end function
***Advance an expansion region one step***
function AdvanceRegion(Region r, Configurations Q)
    if r.AtEdgeEnd() then
        ***Attempt to merge components***
        C ← GetLocalComponents(r.edge.target)
        for all q ∈ Q do
            E ← AttemptConnections(q, C)
            if E ≠ ∅ then
                merge all c ∈ C connected by E
            end if
        end for
        if not merged then
            InitializeRegions(Q, r.edge.target)
        end if
        DeleteRegion(r)
    else
        ***Move to the next position***
        r.center ← r.GetNextSkeletonEdgePoint()
        r.radius ← GetRegionRadius(r.center)
    end if
end function
```

Chapter 3

Methodology

As mentioned previously, although DR-PRM effectively guides roadmap construction using workspace skeletons, it assumes that skeleton vertices lie in locations with sufficient clearance for sampling. As seen in the pseudocode of DR-PRM, expansion regions are created during initialization around skeleton vertices and later selected during the iterative expansion phase. If a skeleton vertex lies too close to an obstacle, the initialization step may fail to generate valid configurations, preventing the creation of usable regions. Similarly, during iteration, the planner may fail to select any expansion region if all regions have insufficient clearance.

To address this limitation, we propose Gradually Improving Guided PRM (GIG-PRM). Instead of discarding low-clearance regions, GIGPRM allows expansion regions to be gradually relocated to nearby feasible space during the iteration phase. By adjusting region locations dynamically, the planner can recover from imperfect skeleton placement while still leveraging the structural guidance provided by the skeleton, as shown in Figure 3.1.

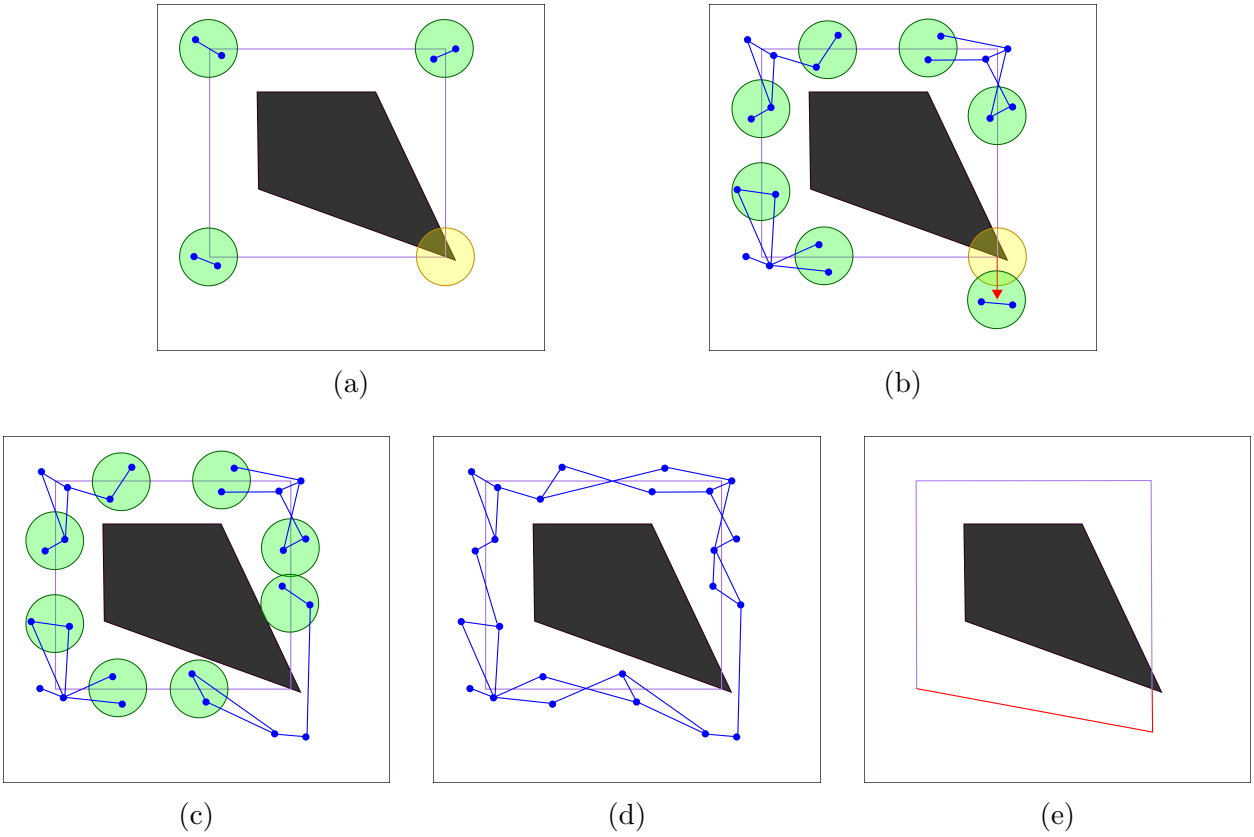


Figure 3.1: Overview of GIGPRM. (a) The workspace skeleton is shown in purple. At the bottom-right corner, the skeleton overlaps the obstacle, so the corresponding yellow region would normally be discarded in DR-PRM due to insufficient clearance. In GIGPRM, this region is preserved and left for later adjustment. The other valid regions are initialized normally and used to form initial connected components. (b) The valid regions advance along the skeleton, while the preserved invalid region is adjusted and moved to a nearby free-space location that no longer overlaps the obstacle, where it can generate valid samples and create a new local component. (c) After the invalid region has been successfully adjusted, it can continue advancing by moving forward along the original skeleton. (d) As expansion proceeds, local components are connected and merged. (e) Finally, GIGPRM outputs an updated skeleton in which the center of the adjusted region is recorded and used as a new skeleton vertex location.

3.1 Algorithm

3.1.1 Initialization

Unlike DR-PRM, which creates a region at a skeleton vertex only when the vertex has positive clearance and can generate at least one valid sample (as shown in Algorithm 3, Line 3-11), GIGPRM skips this clearance checking, allowing regions to be created even if they are initially invalid.

This modification enables GIGPRM to retain structurally meaningful skeleton information that might otherwise be lost. Since regions in GIGPRM can be adjusted or relocated during later iterations, initially invalid regions can be moved into feasible space and then utilized for sampling. By preserving these regions, the planner maintains the connectivity and avoids eliminating potentially important areas of the configuration space.

3.1.2 Iteration

During iteration, GIGPRM modifies how the planner handles two distinct cases: when no region can be selected, and when a specific region is selected but has insufficient clearance.

When no region is selected, GIGPRM selects a random unconnected skeleton edge and attempts to connect its local components(Line 4-12). If one side of the edge has no local component, the planner falls back to directly attempting to merge existing connected components (CCs) in the roadmap using nearest-neighbor local planning(Line 6-10). If the two sides both have local components, ConnectEdgeSegment is called to attempt bridging them(Line 10-11). Once this handling is complete, the iteration returns immediately without performing any region expansion.

When a specific expansion region is selected but its current center lies in or too close to an obstacle which gives it a negative clearance, GIGPRM iterates through the existing regions and attempts to shift one region to a nearby feasible location(Line 19-30). The shifting is performed by incrementally stepping in all possible directions (four directions for

2D settings and 6 directions for 3D settings). The step size is derived from the robot's radius to ensure that each displacement is sufficiently significant relative to the robot's geometry. At each incremental step, we evaluate clearance and reject candidate positions that fall inside obstacles or provide insufficient free space. Once a candidate center with acceptable clearance is identified, the coordinate of the region's center is updated accordingly(Line 24-27).

Whether the region was relocated or already had positive clearance, the planner proceeds to expand the component, which is the same as in DR-PRM: a valid region is probabilistically selected, new samples are generated within the region, and connectivity attempts are made between neighboring components(Line 31-38). Thus, GIGPRM preserves the core dynamic expansion framework of DR-PRM while introducing a region relocation mechanism that improves robustness when the initial skeleton is imperfect.

3.1.3 Implementation Details

Algorithm 3 provides a description of the relocation step. The following describes the implementation details used to determine how the region center is shifted and when the search along a direction terminates. In practice, the region center is not shifted in a single step. Instead, the algorithm incrementally moves the center along each search direction and evaluates the clearance at every step. The implementation allows one such obstacle encounter along a direction, but terminates the search along that direction after a second obstacle hit. This is because the second hit indicates that direction is probably obstacle-dense and it's better to switch to another direction. There is also a limitation on the maximum number of incremental steps along a direction, which prevents the region from drifting too far along undesirable directions.

Once a candidate center satisfies the clearance requirements, the region's center is updated by modifying its offset relative to the original skeleton vertex. This effectively relocates the region without altering the underlying skeleton structure directly. After relocation, the planner immediately attempts to expand the region using the updated center, and then

Algorithm 3 GIGPRM Iterate Procedure

```
1: function ITERATE
2:     region ← SELECTEXPANSIONREGION()
3:     if region = null then
   //Whole-environment fallback
4:         if unconnected edges exist then
5:             edge ← randomly select one unconnected edge
6:             if either side of edge has no local component then
7:                 repeat
8:                     CONNECTCC()
   //Merge roadmap CCs
9:                 until no further CC merges succeed
10:            else
11:                CONNECTEDGESEGMENT(edge)
12:            end if
13:        else
14:            return
15:        end if
16:        return
17:    end if
18:    clearance ← GETREGIONRADIUS(region.center)
19:    if clearance < 0 then
   //Attempt to relocate region
20:        moved ← false
21:        for all direction d do
22:            candidate ← shift center of region along d
23:            newClearance ← GETREGIONRADIUS(candidate)
24:            if candidate in boundary and newClearance ≥ stepSize then
25:                RELOCATECENTER(region, candidate)
26:                record displacement in m_vertexOverride
27:                moved ← true; break
28:            end if
29:        end for
30:    end if
   //Standard DR-PRM expansion
31:    samples ← EXPANDCOMPONENT(region)
32:    if samples ≠ ∅ then
33:        while ADVANCEREGION(region, samples) and ARESAMPLESCOVERED(region, samples) do
34:        end while
35:    end if
36:    if region still exists and region edge not connected then
37:        CONNECTEDGESEGMENT(region edge)
38:    end if
39: end function
```

follows the original DR-PRM structure.

Algorithm 4 Finalize and Output Updated Skeleton

```
1: function FINALIZE
   // same as standard strategies
   ...
   // output updated skeleton file
2:     for all vertex v in out do
3:         if v has an override position then
4:             replace v.position with override value
5:         end if
6:     end for
7:     for all edge e in out do
8:         u ← source vertex position
9:         v ← target vertex position
10:        for all intermediate point k on edge e do
11:            linearly interpolate between u and v
12:            update point k
13:        end for
14:    end for
15:    Write updated skeleton to a new file
16: end function
```

In Finalize(), GIGPRM produces an updated version of the workspace skeleton to reflect any region relocations performed during iteration.

During the iteration phase, whenever a region is moved, the new position is recorded in a data structure (*m_vertexOverride*) that stores the updated coordinates associated with the corresponding skeleton vertices. To do this, the original skeleton is copied into a local variable. The code then iterates through all vertices in the skeleton. If a vertex has a recorded override in *m_vertexOverride*, which stores relocated region endpoints, its position is replaced with the updated coordinates(Algorithm 4, Line 2-6).

After updating the vertex positions, each edge is recomputed in order to generate a skeleton file that is compatible with the updated skeleton structure(Algorithm 4, Line 7-14). The intermediate points along each edge are recalculated by evenly spacing them along the straight line between the updated source and target vertices on the ends of each edge.

Chapter 4

Experiments

We compared the performance of GIGPRM with DR-PRM(Guided-PRM) using skeletons of varying quality across three environments: 3D, Mining, and ZigZag, all of which are motion planning benchmarks from the Parasol Lab.

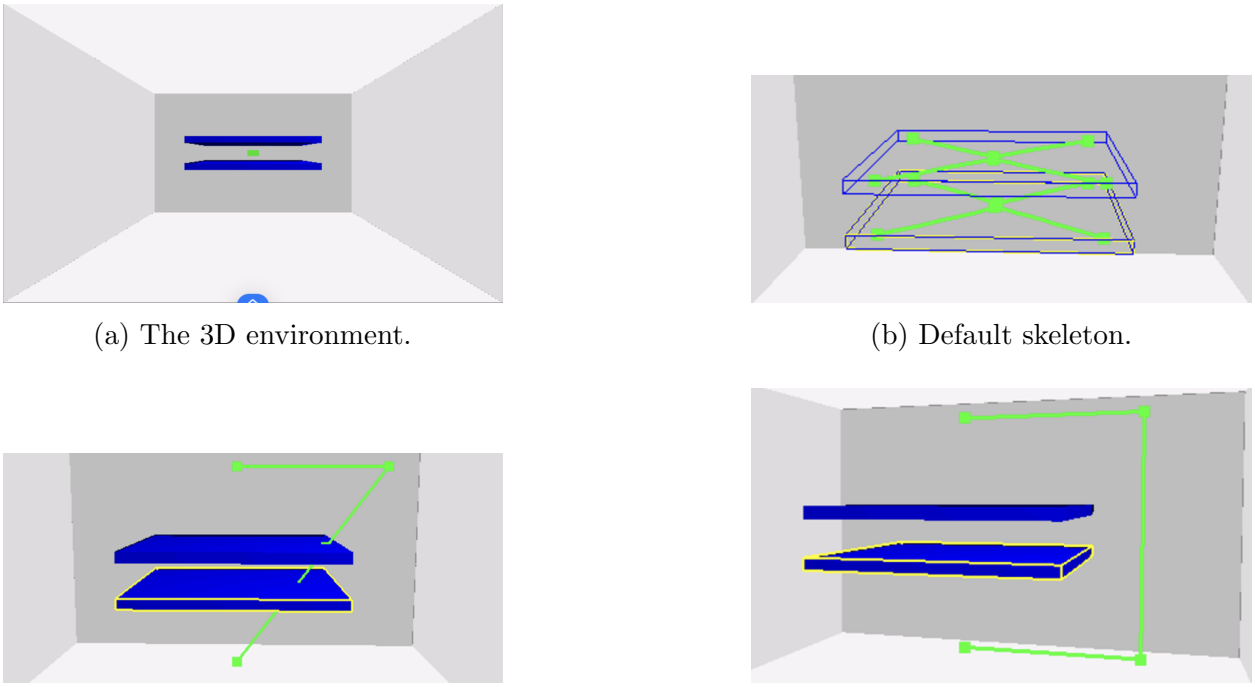
4.1 Experimental Setup

4.1.1 Test Environments

We selected three environments that differ in geometric difficulty and skeleton quality:

- 3D Environment. As shown in Figure 4.1, the 3D environment is a largely open three-dimensional workspace containing two thin obstacles. Three skeletons were used for this environment: a default skeleton, a partially faulty skeleton, and a curated skeleton. The default skeleton, generated entirely inside the obstacles, represents the lowest-quality case and is used to evaluate planner performance when the skeleton is severely flawed. The partially faulty and curated skeletons were both constructed manually; the partially faulty skeleton is derived from the curated one by displacing a single vertex into an obstacle, causing overlap between the skeleton and the obstacle geometry. This configuration is designed to test performance when the skeleton remains structurally

valid but is locally inaccurate. The curated skeleton lies entirely within the free space and serves as the reference standard for comparison.



(a) The 3D environment.

(b) Default skeleton.

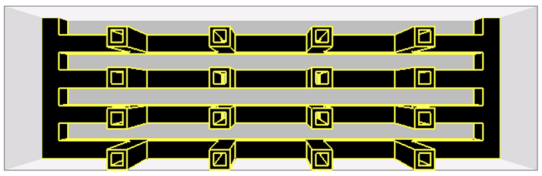
(c) Partially faulty skeleton.

(d) Curated skeleton.

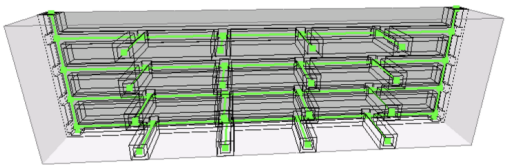
Figure 4.1: The 3D environment and its three skeletons of varying quality. (a) shows the environment; (b)–(d) show the default, partially faulty, and curated skeletons. Obstacles are shown in blue, and skeleton vertices and edges are shown in green.

- Mining Environment. The Mining environment, shown in Figure 4.2, is a three-dimensional workspace consisting of narrow tunnels embedded within obstacles, surrounded by open space that does not contribute to feasible paths between the start and goal configurations. Four skeletons were used for this environment: one curated skeleton and three perturbed variants in which 5, 10, or 15 vertices were randomly displaced from the curated baseline. These variants represent increasing levels of skeleton inaccuracy.

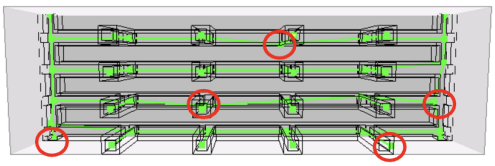
- ZigZag Environment. The ZigZag environment is a two-dimensional workspace whose feasible region is a narrow S-shaped corridor bounded by obstacles. A single skeleton was used, generated automatically by the workspace skeleton constructor in the Parasol



(a) The Mining environment.



(b) Curated skeleton.



(c) Perturbed skeleton.

Figure 4.2: The Mining environment and its skeletons. (a) shows the workspace, consisting of narrow tunnels embedded in obstacles. (b) shows the curated skeleton lying within the free space of the tunnels. (c) shows an example of a perturbed skeleton, in which a subset of vertices have been randomly displaced from the curated baseline; the displaced vertices are highlighted with red circles.

Planning Library. As shown in Figure 4.3, this skeleton contains isolated vertices that are not connected to any edges. This environment therefore evaluates planner performance when the skeleton is disconnected.

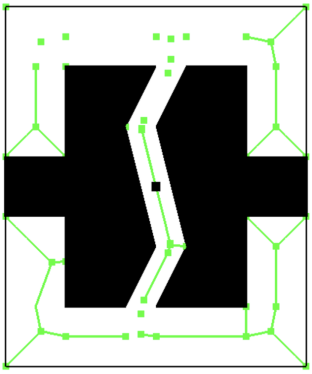


Figure 4.3: The ZigZag environment and its automatically generated skeleton. The feasible region is a narrow S-shaped corridor between obstacles (shown in black). The skeleton, generated by the workspace skeleton constructor in the Parasol Planning Library, contains isolated vertices that are not connected to the rest of the skeleton by any edges.

4.1.2 Metrics

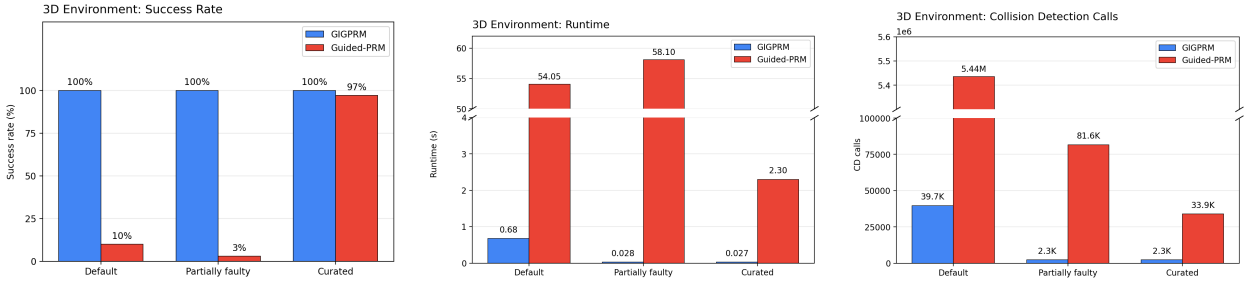
Performance was measured using the following metrics: (i) path-finding rate, as a measure of reliability; (ii) planning time, as a measure of efficiency; and (iii) the number of collision detection (CD) calls, which reflect computational cost.

For each environment–skeleton pair, both planners were executed over 30 independent runs with different random seeds. The path-finding rate is defined as the percentage of runs in which a valid path is found within a fixed time limit. The time limit was set per environment based on its complexity: 60 seconds for the 3D and ZigZag environments, and 200 seconds for the Mining environment. Runs that did not produce a valid path within these limits were considered failures, indicating that the planner could not solve the problem efficiently.

4.2 Results

4.2.1 3D Environment

Figure 4.4 reports the three metrics for the 3D environment under the default, partially faulty, and curated skeletons.



(a) Success rate. (b) Runtime. (c) Collision detection calls.

Figure 4.4: Performance of GIGPRM and Guided-PRM in the 3D environment under three skeleton qualities. Broken y-axes are used in (b) and (c) to keep both small and large values readable.

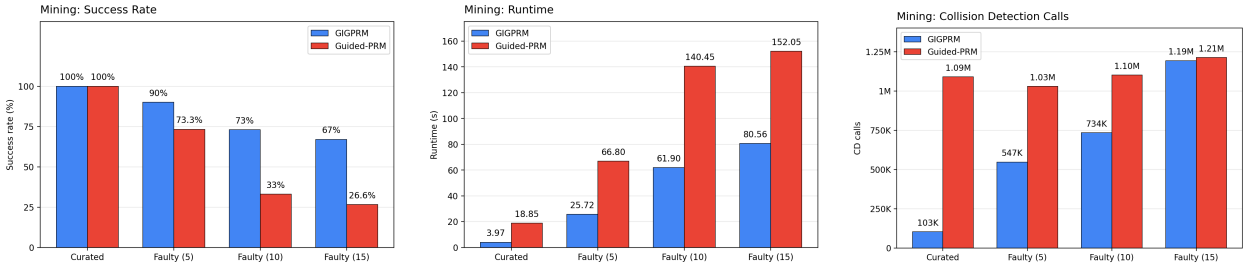
Analysis GIGPRM achieves a 100% path-finding rate on every skeleton, including the default skeleton that lies entirely inside the obstacles. By contrast, Guided-PRM is highly sensitive to skeleton quality: it succeeds on only 10% of runs with the default skeleton and 3% with the partially faulty skeleton, recovering to 97% only when the skeleton is curated. This indicates that GIGPRM is largely robust to inaccuracies in the skeleton, whereas Guided-PRM relies on the skeleton being medially centered without mistake.

The runtime and CD call results reinforce this observation. With the default skeleton, Guided-PRM spends nearly the entire 60-second budget (54.05 s) and calls over five million CD calls, while GIGPRM finishes in 0.68 s with about 40k CD calls. With the partially faulty skeleton, Guided-PRM degrades further to 58.10 s, while GIGPRM completes in roughly 0.03 s. Even in the best case (curated skeleton), GIGPRM is approximately two orders of magnitude faster (0.027 s versus 2.30 s) and uses fewer than 7% of the CD calls required by Guided-PRM. Together, these results show that GIGPRM not only tolerates poor skeletons

but also achieves significant efficiency improvements when the skeleton is well-formed.

4.2.2 Mining Environment

Figure 4.5 shows results for the Mining environment, where the curated skeleton is progressively perturbed by randomly displacing 5, 10, or 15 vertices.



(a) Success rate. (b) Runtime. (c) Collision detection calls.

Figure 4.5: Performance of GIGPRM and Guided-PRM in the Mining environment as the skeleton is increasingly perturbed. Skeletons are ordered from highest to lowest quality.

Analysis On the curated skeleton, both planners achieve a 100% path-finding rate, so the comparative behavior only emerges as the skeleton degrades. As the number of perturbed vertices grows from 5 to 15, the success rate of Guided-PRM drops sharply from 73.3% to 26.6%, while GIGPRM declines more gradually from 90% to 67%. This gentler decline is consistent with the conclusion drawn in the 3D environment: GIGPRM degrades gracefully under skeleton inaccuracy, whereas Guided-PRM is brittle.

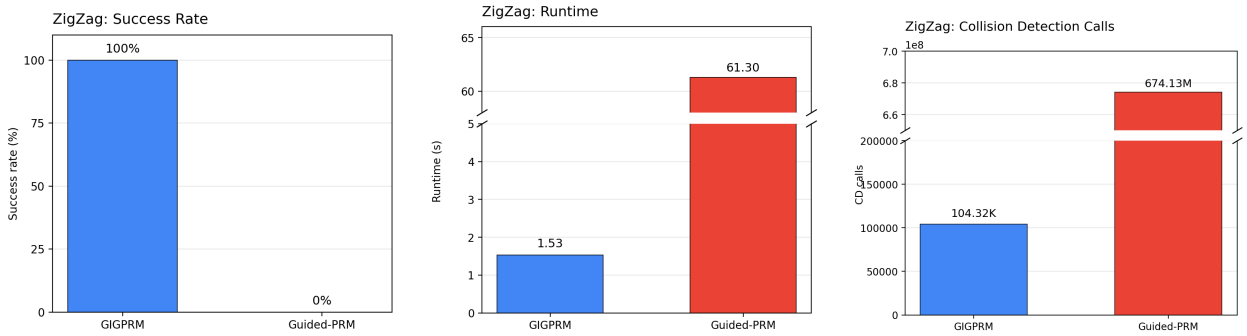
In runtime, GIGPRM is consistently around half as expensive as Guided-PRM at every perturbation level (e.g., 25.72 s versus 66.80 s at 5 perturbed vertices, and 80.56 s versus 152.05 s at 15). The runtime grows for both planners as the skeleton worsens, which is same as expected: a less accurate skeleton leads to more sampling and validation work in the narrow tunnels.

The CD call results reveal a more nuanced trend. For high-quality skeletons, GIGPRM uses an order of magnitude fewer CD calls than Guided-PRM (103k versus 1.09M on the curated skeleton). However, the gap narrows steadily as the skeleton degrades, and at

15 perturbed vertices the two planners issue almost the same number of CD calls (1.19M versus 1.21M).

4.2.3 ZigZag Environment

Figure 4.6 reports results for the ZigZag environment, in which the only available skeleton is automatically generated and contains isolated, disconnected vertices.



(a) Success rate. (b) Runtime. (c) Collision detection calls.

Figure 4.6: Performance of GIGPRM and Guided-PRM on the ZigZag environment with the automatically generated, partially disconnected skeleton.

Analysis The ZigZag environment provides the clearest result of the two strategies, because the only skeleton available is structurally incomplete. GIGPRM achieves a 100% path-finding rate, while Guided-PRM fails on every run within the 60-second time limit (0% success). The runtime numbers reflect the same outcome: GIGPRM solves the problem in 1.53 s, whereas Guided-PRM exhausts the full budget and is recorded at 61.3 s on average. The CD call counts are even more dramatic, with GIGPRM issuing roughly 104k calls and Guided-PRM issuing about 674M, a difference of more than three orders of magnitude.

This outcome highlights the central distinction between the two strategies. When the skeleton is disconnected, Guided-PRM cannot bridge across the missing edges, and the planner is unable to connect to the regions on the isolated skeleton vertices. GIGPRM, by contrast, can fall back on its own mechanism when the skeleton fails to provide useful guidance, allowing it to connect components directly.

4.3 Discussion

According to the results, we can conclude about how GIGPRM behaves under varying skeleton quality. It achieves both higher reliability and lower computational cost than Guided-PRM whenever the skeleton is well positioned, and it remains solvable in cases where Guided-PRM fails. The path-finding rate of GIGPRM stays at or near 100% in simple environments such as 3D and ZigZag, regardless of skeleton quality, and degrades only gradually in more complex environments like the Mining, as the skeleton is increasingly perturbed.

One observation is that GIGPRM does not reach a 100% path-finding rate in the Mining environment under any of the perturbed skeletons, even though it does so in the 3D and ZigZag environments. The reason lies in the structure of the Mining geometry. The tunnels are bounded by thin walls that have free space on both sides, and this geometry works poorly with GIGPRM's region adjusting mechanism. When a skeleton vertex is perturbed into a thin wall, GIGPRM attempts to displace nearby samples back into free space, but because the obstacle has free space on both sides, the planner cannot determine the correct direction of displacement, as shown in Figure 4.7. As a result, a fraction of the recovery attempts move the region to the wrong side of the obstacle that does not connect to the goal.

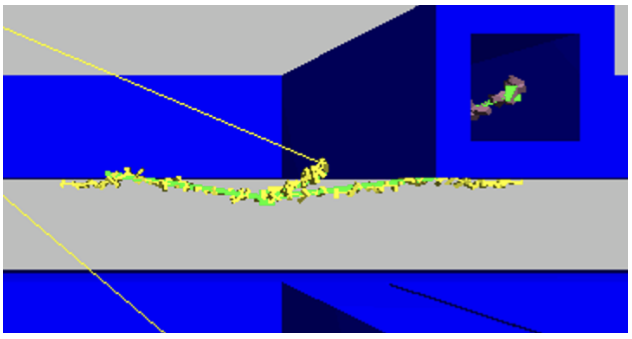


Figure 4.7: A failure case in the Mining environment. The blue regions are obstacles, and the green graph is the skeleton. Near the center of the figure, a skeleton vertex has been displaced outside the obstacle by GIGPRM's region adjustment. As a consequence, the yellow local component generated at this vertex is also placed outside the tunnel, where it cannot connect to the components growing inside the tunnel and is therefore stranded with respect to the start and goal configurations.

However, the Mining results do not indicate a failure of GIGPRM's strategy. Instead, they reveal a specific geometric configuration in which the region recovery alone cannot resolve the direction of motion, and addressing this is left as future work.

Chapter 5

Conclusion

This thesis addressed two core research questions. The first asked how skeleton-guided planning approaches can be adapted to remain reliable and efficient when workspace skeletons are imperfect. GIGPRM answers this by treating the skeleton as an adjustable guidance structure rather than a fixed constraint. By preserving invalid regions and incrementally shifts them toward nearby feasible space during iteration, GIGPRM gradually refines skeleton guidance during roadmap construction without requiring the skeleton to be accurate from the start. The second question asked whether sampling in configuration space can be used to adjust workspace skeletons so they better reflect true connectivity and clearance. The results confirm that this is both possible and effective.

We compared GIGPRM against DR-PRM across three benchmark environments from the Parasol Planning Library, including 3D, Mining, and ZigZag, using skeletons from fully curated to entirely faulty, partially perturbed, and structurally disconnected.

The experiments support three main findings. First, GIGPRM is substantially more robust to skeleton quality than DR-PRM, achieving a high path-finding rate across all skeleton conditions where DR-PRM collapses. Second, GIGPRM is more efficient than DR-PRM regardless of skeleton quality, completing planning queries with consistently lower runtime and fewer collision detection calls. Third, although GIGPRM's performance decreases in

environments containing thin walls flanked by free space on both sides, it remains reliable relative to DR-PRM, indicating that even in geometrically adverse cases its recovery mechanism continues to provide a meaningful advantage.

The core innovation of GIGPRM is its skeleton-fixing mechanism, which relaxes a restriction shared by previous skeleton-guided strategies, which is the assumption that the skeleton is medially centered within the free space. By explicitly correcting samples around displaced or invalid skeleton vertices, GIGPRM is able to use skeletons that would render guided strategies ineffective. This relaxation makes skeleton-guided planning practical in settings where the skeleton is generated automatically from the workspace and cannot be guaranteed to be accurate, complete, or properly aligned with the medial axis.

5.1 Future Work

The current region adjustment strategy attempts to displace samples in all possible directions and retains whichever displacement yields a valid configuration. While effective in most cases, this brute-force approach becomes inefficient and unreliable in environments containing thin walls with free space on both sides, as observed in the Mining experiments. Developing a smarter region adjustment strategy that can identify the correct direction more efficiently is a promising direction for future work.

Besides, the analysis in this thesis has been entirely empirical. While the experimental results are consistent with probabilistic completeness, a formal proof has not been established. Establishing such a proof, and characterizing the conditions on the skeleton-fixing mechanism under which completeness holds, should be a future direction of this thesis.

Finally, all experiments in this thesis were conducted in synthetic, hypothesized environments. Although a more realistic benchmark environment was available, the geometry adjustments required to make it compatible with GIGPRM could not be completed within the scope of this project, so it could not be included in the evaluation. A natural extension of

this work is therefore to validate GIGPRM on environments derived from real-world settings, which would provide a stronger test of its practical applicability and robustness beyond the controlled benchmarks used here.

Bibliography

[1] Harry Blum. "A transformation for extracting new descriptions of shape". In: *Models for the perception of speech and visual form* (1967), pp. 362–380.

[2] V. Boor, M.H. Overmars, and A.F. van der Stappen. "The Gaussian sampling strategy for probabilistic roadmap planners". In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. Vol. 2. 1999, 1018–1023 vol.2. DOI: 10.1109/ROBOT.1999.772447.

[3] Jory Denny, Read Sandström, Andrew Bregger, and Nancy M. Amato. "Dynamic Region-biased Rapidly-exploring Random Trees". In: *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*. Ed. by Ken Goldberg, Pieter Abbeel, Kostas Bekris, and Lauren Miller. Cham: Springer International Publishing, 2020, pp. 640–655. ISBN: 978-3-030-43089-4. DOI: 10.1007/978-3-030-43089-4_41. URL: https://doi.org/10.1007/978-3-030-43089-4_41.

[4] Songyi Dian, Juntong Liu, Guofei Xiang, and Xingxing You. "Bridge Points Guided Neural Motion Planning in Complex Environments with Narrow Passages". In: *Sensors* 26.5 (2026). ISSN: 1424-8220. DOI: 10.3390/s26051582. URL: https://www.mdpi.com/1424-8220/26/5/1582.

[5] Andreas Fabri and Sylvain Pion. "CGAL: The computational geometry algorithms library". In: *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. 2009, pp. 538–539.

[6] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. "A review of motion planning techniques for automated vehicles". In: *IEEE Transactions on intelligent transportation systems* 17.4 (2015), pp. 1135–1145.

[7] David Hsu, Lydia E. Kavraki, Jean-Claude Latombe, Rajeev Motwani, and Stephen Sorkin. "On finding narrow passages with probabilistic roadmap planners". In: *Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics on Robotics: The Algorithmic Perspective: The Algorithmic Perspective*. WAFR '98. Houston, Texas, USA: A. K. Peters, Ltd., 1998, pp. 141–153. ISBN: 1568810814.

[8] Brian Ichter, James Harrison, and Marco Pavone. "Learning sampling distributions for robot motion planning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 7087–7094.

[9] N. Jouandeau and A. Ali Cherif. "Fast approximation to gaussian obstacle sampling for randomized motion planning". In: *IFAC Proceedings Volumes* 37.8 (2004). IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal, 5-7 July 2004, pp. 591–596. ISSN: 1474-6670. DOI: https://doi.org/10.1016/S1474-6670(17)32042-6. URL: https://www.sciencedirect.com/science/article/pii/S1474667017320426.

[10] Rahul Kala. "Increased visibility sampling for probabilistic roadmaps". In: *2018 IEEE international conference on simulation, modeling, and programming for autonomous robots (SIMPAR)*. IEEE. 2018, pp. 87–92.

[11] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894. DOI: 10.1177/0278364911406761. URL: https://doi.org/10.1177/0278364911406761.

[12] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580. DOI: 10.1109/70.508439.

[13] Jean-Claude Latombe. *Robot motion planning*. Vol. 124. Springer Science & Business Media, 2012.

[14] Steven M. LaValle. "Rapidly-exploring random trees : a new tool for path planning". In: *The annual research report* (1998). URL: https://api.semanticscholar.org/CorpusID:14744621.

[15] Erion Plaku, Lydia E Kavraki, and Moshe Y Vardi. "Motion planning with dynamics by a synergistic combination of layers of planning". In: *IEEE Transactions on Robotics* 26.3 (2010), pp. 469–482.

[16] Markus Rickert, Oliver Brock, and Alois Knoll. "Balancing exploration and exploitation in motion planning". In: *2008 IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 2812–2817.

[17] Samuel Rodriguez, Xinyu Tang, Jyh-Ming Lien, and Nancy M Amato. "An obstacle-based rapidly-exploring random tree". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE. 2006, pp. 895–900.

[18] Read Sandström, Diane Uwacu, Jory Denny, and Nancy M. Amato. "Topology-Guided Roadmap Construction With Dynamic Region Sampling". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6161–6168. DOI: 10.1109/LRA.2020.3010487.

[19] T. Siméon, J.-P. Laumond, and C. Nissoux. "Visibility-based probabilistic roadmaps for motion planning". In: *Advanced Robotics* 14.6 (2000), pp. 477–493. DOI: 10.1163/156855300741960. eprint: https://doi.org/10.1163/156855300741960. URL: https://doi.org/10.1163/156855300741960.

[20] Zheng Sun, D. Hsu, Tingting Jiang, H. Kurniawati, and J.H. Reif. "Narrow passage sampling for probabilistic roadmap planning". In: *IEEE Transactions on Robotics* 21.6 (2005), pp. 1105–1115. DOI: 10.1109/TRO.2005.853485.

[21] Jakub Szkandera, Ivana Kolingerová, and Martin Maňák. "Narrow Passage Problem Solution for Motion Planning". In: *Computational Science – ICCS 2020*. Ed. by Valeria V. Krzhizhanovskaya, Gábor Závodszky, Michael H. Lees, Jack J. Dongarra, Peter M. A. Sloot, Sérgio Brissos, and João Teixeira. Cham: Springer International Publishing, 2020, pp. 459–470. ISBN: 978-3-030-50371-0.

[22] Andrea Tagliasacchi, Ibraheem Alhashim, Matt Olson, and Hao Zhang. "Mean curvature skeletons". In: *Computer Graphics Forum*. Vol. 31. 5. Wiley Online Library. 2012, pp. 1735–1744.

[23] Diane Uwacu, Ananya Yammanuru, Marco Morales, and Nancy M Amato. "Hierarchical planning with annotated skeleton guidance". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 11055–11061.

[24] Diane Uwacu, Ananya Yammanuru, Keerthana Nallamotu, Vasu Chalasani, Marco Morales, and Nancy M Amato. "HAS-RRT: RRT-Based Motion Planning Using Topological Guidance". In: *IEEE Robotics and Automation Letters* (2025).