GuILD: Guided Incremental Local Densification for Accelerated Sampling-based Motion Planning

Rosario Scalise∗1, Aditya Mandalika∗2, Brian Hou∗1, Sanjiban Choudhury3 and Siddhartha S. Srinivasa1

Abstract—Sampling-based motion planners rely on incremental densification to discover progressively shorter paths. After computing feasible path ξ between start \( x_s \) and goal \( x_t \), the Informed Set (IS) prunes the configuration space \( \mathcal{X} \) by conservatively eliminating points that cannot yield shorter paths. Densification via sampling from this Informed Set retains asymptotic optimality of sampling from the entire configuration space. For path length \( c(\xi) \) and Euclidean heuristic \( h \), \( IS = \{ x | x \in \mathcal{X}, h(x_s, x) + h(x, x_t) \leq c(\xi) \} \).

Relying on the heuristic can render the IS especially conservative in high dimensions or complex environments. Furthermore, the IS only shrinks when shorter paths are discovered. Thus, the computational effort from each iteration of densification and planning is wasted if it fails to yield a shorter path, despite improving the cost-to-come for vertices in the search tree. Our key insight is that even in such a failure, shorter paths to vertices in the search tree (rather than just the goal) can immediately improve the planner’s sampling strategy. Guided Incremental Local Densification (GuILD) leverages this information to sample from Local Subsets of the IS. We show that GuILD significantly outperforms uniform sampling of the Informed Set in simulated \( \mathbb{R}^5 \), \( SE(2) \) environments and manipulation tasks in \( \mathbb{R}^3 \).

I. INTRODUCTION

Sampling-based algorithms have shown tremendous success in solving complex high-dimensional robot motion planning problems. To achieve asymptotic-optimality, these algorithms incrementally sample the robot’s configuration space to continually improve the shortest path in an anytime manner [1]–[6]. A discrete sampling-based approximation to the continuous configuration space often yields an initial feasible path quickly. However, converging to the optimal path is typically slow: many configuration samples are needed to improve the discrete approximation (i.e., cover the space more densely) and discover a shorter path. Replanning with new samples can be accomplished efficiently [7], so the computational bottleneck is sampling in regions that result in a better path.

Our focus is on accelerating convergence to the optimal path, assuming that an initial feasible path has already been discovered. The current best path \( \xi \) between the start \( x_s \) and target \( x_t \) defines an Informed Set \( \mathcal{X}_{IS} \subseteq \mathcal{X} \) that contains only states that can yield shorter paths:

\[
\mathcal{X}_{IS} = \{ x \ | \ x \in \mathcal{X}, h(x_s, x) + h(x, x_t) \leq c(\xi) \}, \quad (1)
\]

where \( c(\xi) \) is the cost of the current path and \( h \) is an admissible heuristic [8]. For problems minimizing path length in \( \mathbb{R}^n \), the Informed Set with the Euclidean distance heuristic is an \( n \)-dimensional prolate hyperspheroid \( \mathcal{E}_{IS} \), parameterized by foci \( x_s \) and \( x_t \) and transverse axis diameter \( c(\xi) \) (i.e., a generalized ellipse as seen in Fig. 1a). Sampling from \( \mathcal{E}_{IS} \) preserves the asymptotic optimality guarantee of sampling from the entire configuration space \( \mathcal{X} \).

In practice, however, the Informed Set often provides limited sample efficiency improvement. Since the measure (\( n \)-volume) of the Informed Set \( \lambda(\mathcal{E}_{IS}) \) is a function of the current solution cost, a high cost solution may yield a large Informed Set with comparable (or greater) measure to the full state space \( \lambda(\mathcal{X}) \). Paradoxically, to reap the greatest benefits of sampling from the Informed Set, the planner must already have a path of sufficiently small cost. This is a particular challenge for planning problems in high dimensions or cluttered environments where feasible paths likely have high cost. Furthermore, the computational effort from each iteration of densification and planning is wasted if it fails to yield a shorter path. Until a shorter path is discovered, \( \mathcal{E}_{IS} \) (and thus the sampling distribution) remains unchanged.

Our key insight is that even in such a failure, we can open the black box of the planning algorithm to immediately improve the planner’s sampling strategy. With Guided Incremental Local Densification (GuILD), shorter paths to any vertex in the search tree can guide further sampling. GuILD introduces the idea of a beacon, a vertex in the search tree that decomposes the original sampling/planning problem into two smaller subproblems (Fig. 1b). Much like the Informed Set between the start and target, the beacon induces...
Local Subsets with (i) start and beacon as foci and (ii) beacon and target as foci. GuILD leverages improvements to the search tree to adapt the Local Subsets and converge to the optimal path with fewer samples (Fig. 2).

Generating samples from the continuous configuration space that will accelerate convergence to the optimal path is a challenging problem. Most approaches extract geometric information or low-dimensional structure from the robot workspace to bias the sampling distribution [9]–[12]. However, these methods can be computationally expensive and sensitive to covariate shift. GuILD simplifies this into a discrete problem of identifying beacons that yield focused samples. Local Subsets can be sampled efficiently, and the distribution is biased online by leveraging cost information already captured in the search tree.

GuILD only uses vertices in the search tree to construct Local Subsets. While any configuration in the Informed Set can be used to define subproblems, the lower bounds from the admissible heuristic are insufficient for constraining the two transverse axis diameters. In contrast, the current cost-to-come for vertices in the search tree provides an additional upper bound that uniquely defines the Local Subsets.

We make the following contributions:

- We introduce GuILD, an incremental densification framework that effectively leverages search tree information to focus sampling.
- We compare theoretical properties of the Local Subsets that GuILD samples to the Informed Set.
- We propose several BEACONSELECTOR strategies for GuILD, including an adversarial bandit algorithm.
- We show experimentally that regardless of the BEACONSELECTOR, GuILD outperforms the state-of-the-art Informed Set densification baseline across a range of planning domains. In particular, GuILD yields modest improvements in simpler planning domains and excels in domains with difficult-to-sample homotopy classes.

II. RELATED WORKS

Sampling-based algorithms construct roadmaps [13] or trees [14]–[16] by sampling configurations and connecting nearest neighbors to compute a collision-free shortest path. Sampling-based algorithms for optimal motion planning incrementally densify the configuration space to determine shorter paths in an anytime manner [1], [3]–[6], [17] guaranteeing asymptotic optimality. However, their convergence is typically slow, especially in higher dimensions and complex environments with narrow passages; a uniform sampler requires \( O(\delta^{-d}) \) configuration samples to discover the optimal path with \( \delta \)-clearance in a \( d \)-dimensional space, which can be computationally prohibitive [16].

This motivated literature that focused on improving the sampling strategy. A popular approach is to extract information from the workspace to bias sampling. One class of algorithms samples between regions of collision to identify narrow passages [9], [18]–[23], or samples close to obstacles to compute high quality paths that follow contours of the obstacles [24]–[26]. This reduces the number of samples required to compute a solution, at the expense of computationally-intensive geometric tests to accept samples, resulting in large planning times. Another class of algorithms [27], [28] instead bias sampling by initially solving a coarse approximation of the workspace, and the solution informs subsequent sampling. These approaches, however, can be used in parallel with GuILD to sample non-uniformly within the Local Subsets.

Recently, to alleviate the issues with approaches relying on explicit geometric tests, there has been effort in determining low-dimensional structure in the workspace to bias sampling. Generative modeling tools have been applied to learn sampling distributions, conditioned upon the current planning problem and the obstacle distribution [10]–[12], [29]–[31]. These approaches do not explicitly reason about the Informed Set to provide optimality guarantees, and fail to robustly sample bottleneck regions in complex environments. As a result, the benefits are limited to environments that are similar to those reflected in the training data. In contrast,
**GuILD** makes no assumption about the environment structure. However, these learning-based approaches share with **GuILD** the underlying idea of identifying critical samples and can be leveraged to inform beacon selection.

A different class of algorithms adapt sampling online as planning progresses. **GuILD** best fits in this class of algorithms that bias sampling with the state of the planning algorithm. **GuILD** makes no assumption about the environment structure. However, these learning-based approaches share with **GuILD**. Each vertex \( v \in T \) is associated with a parent vertex, a cost-to-come \( g(v) \) via that parent, and a consistent cost-to-go.

Informed Set (IS) excludes states for which this lower bound exceeds the best solution cost \( c(\xi) \). Unlike sampling the entire state space \( X_{free} \), sampling from the IS automatically excludes states that cannot be part of a lower-cost path [8].

Euclidean distance, an admissible heuristic for path length, admits a concise geometric description of the IS: an \( n \)-dimensional prolate hyperspheroid \( E_{IS} = E(x_s, x_t, c(\xi)) \) with foci \( x_s, x_t \) and transverse axis diameter \( c(\xi) \).

**Algorithm 1: Informed Optimal SBMP**

**Input:** start \( v_s \), goal \( v_t \)

**Output:** \( \{\xi_1, \xi_2, \cdots\} \) s.t. \( c(\xi_{i+1}) < c(\xi_i) \)

1. Initialize best solution cost: \( c(\xi) \leftarrow \infty \)
2. Initialize edge-implicit graph: \( G \leftarrow \{v_s, v_t\} \)
3. repeat
   4. Densify: \( G \leftarrow \text{Sample}(v_s, v_t, c(\xi)) \) (⋆)
   5. Compute shortest path: \( \xi \leftarrow \text{Search}(v_s, v_t, G) \)
   6. if \( c(\xi) = g(v_t) < c(\xi) \) then
      7. Update best solution cost: \( c(\xi) \leftarrow g(v_t) \)
      8. Emit current solution \( \xi \)
4. until forever;

Prolate hyperspheroids can be sampled analytically, rather than via rejection sampling [8].

Sampling from the IS is a sufficient condition to converge to \( \xi^* \). The improved efficiency can be characterized by comparing the measure (n-volume) \( \lambda(E_{IS}) \) to the measure of the full state space \( \lambda(X) \). When \( \lambda(E_{IS}) < \lambda(X) \), the IS can yield significant improvements on sample efficiency. However, when the initial solution has high path length (e.g., due to a cluttered environment), \( \lambda(E_{IS}) \) may instead be closer to—or even larger than—\( \lambda(X) \). This issue is exacerbated because the IS only shrinks when shorter paths are found. As a result, each iteration of densification and search that fails to find a shorter path does not affect the state sampling distribution. In the next section, we introduce a new strategy that leverages this previously-wasted computational effort.

**IV. GUIDED INCREMENTAL LOCAL DENSIFICATION**

We present **GuILD**, an incremental densification framework that leverages partial search information to focus sampling within the IS. If an iteration of densification and planning has failed to improve the solution cost, what information is there for **GuILD** to take advantage of?

Our key insight is to open the black box of the underlying search algorithm. Although the iteration may not have found a shorter path to the goal, new shorter paths to other vertices in the search tree can immediately improve the sampling strategy (Alg. 1, Line 4).

**Guiling Densification with Search Tree Information**

During search (Alg. 1, Line 5), the algorithm internally expands vertices in \( G \) to construct a search tree \( T \). Each vertex \( v \in T \) is associated with a parent vertex, a cost-to-come \( g(v) \) via that parent, and a consistent cost-to-go.
Algorithm 2: GuILD

Input: beacon set $B$, search tree $T$, graph $G$
Output: densified graph $G$

1 if a solution does not yet exist then
2 \[ G \leftarrow \text{UniformSample}(X_{\text{free}}) \]
3 else
4 \[ b \leftarrow \text{BEACONSELECTOR}(B) \]
5 \[ E_{LS} \leftarrow E(v_x, b, g(b)) \cup E(b, v_t, c(\xi) - g(b)) \]
6 \[ G \leftarrow \text{UniformSample}(E_{LS}) \]
7 return $G$

Algorithm 3: Candidate BEACONSELECTORS

Input: beacon set $B$

1 Function InformedSet
2 return $v_s$;
3 Function Uniform
4 return $b \sim \mathcal{U}(B)$;
5 Function Greedy
6 return $\arg\max_{b \in B} w(b)$ \Comment{Eq. 4}
7 Function Bandit
8 return EXP3($B$) \Comment{implemented as in [40]}

Fig. 3: Local Subsets $E_{LS}$ (green) are defined by a beacon $b$, as well as the cost-to-come on the search tree $g(b)$ and the current best solution cost $c(\xi)$.

heuristic $h(v, v_t)$. When new states are sampled and added to $G$, they may also be added to $T$. Other vertices may then discover that their cost-to-come would be reduced by updating their parent to this new vertex [1], [4]. While the IS only shrinks when these changes propagate all the way to $v_t$, GuILD leverages any cost-to-come improvement to adaptively guide densification.

GuILD introduces the idea of a beacon: a vertex in the search tree that decomposes the original sampling/planning problem into two smaller subproblems (Fig. 3). We define the Local Subsets (LS) induced by beacon $b$ to be the union of two prolate hyperspheres, with foci $(v_x, b)$ and foci $(b, v_t)$. The start-beacon set has transverse axis diameter $g(b)$, only including points that can improve the cost-to-come from $v_x$ to $b$. Given the current shortest subpath to $b$ and its cost-to-come, the beacon-target set only includes points that can extend that subpath and improve the solution cost by setting the transverse axis diameter to $c(\xi) - g(b)$.

GuILD adapts $E_{LS}$ after each iteration of densification and planning. When $g(b)$ is reduced, the start-beacon set shrinks and the target-beacon set expands. Surprisingly, this expansion is actually desirable: if the best solution cost remains the same and the beacon’s cost-to-come is reduced, there is a larger path length budget that can be expended between the beacon and target that would still yield a shorter path overall (Fig. 2).

We summarize the GuILD framework in Alg. 2, which replaces the starred line of Alg. 1. The BEACONSELECTOR (Alg. 2, Line 4) is a function that chooses a beacon to guide local densification. GuILD samples uniformly from $E_{LS}$ (Line 6) using the same analytic strategy proposed for $E_{IS}$ [39]. GuILD must sample from the IS with nonzero probability to retain asymptotic optimality. This is achieved by including $v_s$ in the beacon set.

In Alg. 3, we present some candidate beacon selectors that we evaluate in Section V. The InformedSet beacon selector recovers the behavior of sampling from the IS by choosing $v_s$ as the beacon. Uniform uniformly samples from the beacon set $B$. The Greedy beacon selector aims to select the beacon with the maximum possible improvement in path length, while minimizing the measure of the set that needs to be sampled. It takes the ratio of these two quantities:

\[ w(b) = \frac{c(\xi) - h(v_x, b) - h(b, v_t)}{\lambda(E_{LS})}. \] (4)

Finally, the Bandit beacon selector implements the EXP3 adversarial bandit algorithm [40]. The reward function is the fractional improvement in the path length

\[ r(b) = \frac{c(\xi_{i-1}) - c(\xi_i)}{c(\xi_{i-1})}. \] (5)

An adversarial bandit algorithm is necessary because this reward function is nonstationary.

Properties of Local Subsets

First, we guarantee that GuILD only samples configurations that can improve the current path length by showing that both sets in $E_{LS}$ are subsets of the IS (Theorem IV.1). Then, we show that the measure (n-volume) of $E_{LS}$ is upper-bounded by that of the IS (Theorem IV.2), so choosing an appropriate beacon will allow GuILD to densify the space more efficiently.

Theorem IV.1. Given the current best solution cost $c(\xi)$, a beacon $b \in T$ previously expanded with cost-to-come $g(b)$, we have that $E(v_x, b, g(b)) \cup E(b, v_t, c(\xi) - g(b)) \subseteq E_{IS}$.

Theorem IV.2. Given the current best solution cost $c(\xi)$, a beacon $b \in T$ previously expanded with cost-to-come $g(b)$, we have $\lambda(E_{LS}) \leq \lambda(E_{IS})$.

V. EXPERIMENTS

We evaluate GuILD on an array of planning problems to characterize the proposed beacon selectors (Fig. 4).

In the $\mathbb{R}^2$ environments, the task is to plan from the bottom left corner to the top right corner (Forest, TwoWall), or the bottom right corner (Trap). A forest of obstacles is randomly placed throughout each environment. The easier Forest environment only has these random obstacles; as
a result, there are many different homotopy classes that will produce near-optimal paths. TwoWall and Trap introduce large obstacles with narrow passages that must be crossed to produce near-optimal paths. Discovering these passages typically requires the space to be sampled very densely. At the lower sampling resolutions resulting from fewer initial graph samples, there are many suboptimal homotopy classes that are more easily sampled and discovered. Therefore, initial paths will have high cost and $\lambda(\mathcal{E}_{IS})$ will be much larger than $\lambda(\mathcal{X})$. These more challenging scenarios highlight the limitations of IS densification.

After characterizing the behavior of GuILD with $\mathbb{R}^2$ environments, we consider higher-dimensional planning problems. In SE2Maze, a benchmark environment from OMPL [41], the task is to navigate a car through a maze. In HERB Shelf, a 7-DOF manipulator [42] is tasked with moving its end-effector to pick an object from a bookshelf.

**Evaluation Metrics and Hypotheses**

We make three hypotheses that characterize the incremental densification performance of GuILD. Sampling from Local Subsets incurs negligible overhead relative to sampling from the Informed Set; both rely on the same analytic sampling primitive for prolate hyperspheroids. To make these results independent of our computing environment, we describe these hypotheses in terms of samples drawn. We compare GuILD to BIT*, which is equivalent to GuILD with the Informed Set BEACONSELECTOR (Alg. 3).

The first metric we consider is the Sample Efficiency of incremental densification: how many samples must be drawn before the optimal SBMP algorithm converges to the cost of the optimal path? We determine this minimum cost $c(\xi^*)$ by running with a large timeout (1 min. for $\mathbb{R}^2$ problems, 5 min. for higher-dimensional planning problems).

**H1** GuILD will require fewer samples to converge to the minimum solution cost than the Informed Set.

Next, to understand the performance of the optimal SBMP algorithm over time, we consider the Convergence Percentage and Normalized Path Cost as a function of samples. Convergence Percentage is the fraction of trials where the planner had converged to the optimal path cost, with that number of samples. Normalized Path Cost divides the cost of the current best solution by the optimal path cost $c(\xi^*).$

**H2** For a fixed sample budget, GuILD will have a higher Convergence Percentage than the Informed Set.

**H3** For a fixed sample budget, GuILD will have lower Normalized Path Cost than the Informed Set.

**Results**

To test these hypotheses, we implement the BEACONSELECTORS described in Section IV. We construct an initial set of beacons $\mathcal{B}$ by sampling states from a low-discrepancy sequence [43]. In our implementation, no new beacons are added to $\mathcal{B}$, and a beacon $b \in \mathcal{B}$ is considered by BEACONSELECTOR only if $b \in \mathcal{E}_{IS}$. We run 100 random trials for each pair of algorithm and environment.

To understand convergence across the random trials, we plot the Convergence Percentage as a function of the number of samples (Fig. 5). To support **H2**, we would expect the IS baseline curve to remain below the GuILD curves. We find this to be the case: on most environments, sampling from the IS results in a lower Convergence Percentage for a fixed sample budget. However, on a third of the TwoWall trials, the Greedy heuristic causes over-sampling in regions that ultimately do not yield the optimal path.

For each environment and instantiation of GuILD, Fig. 5 also shows Bonferroni-corrected nonparametric confidence intervals on the median Sample Efficiency; Sample Efficiency is not normally distributed. We find that in almost all instances, GuILD focuses sampling more efficiently than sampling from the IS, supporting **H1**. In general, the three GuILD selectors achieve comparable Sample Efficiency, suggesting that the key to their success is sampling from Local Subsets. However, the adversarial Bandit selector most consistently ranks among the best performing BEACONSELECTORS, while Uniform and Greedy were less efficient on one environment.

While **H1** shows that GuILD ultimately converges faster to the optimal cost, we would also like to characterize the rate of convergence. For each planning problem, we plot the median Normalized Path Cost across the trials to understand how path length is reduced over time (Fig. 6). Sharp drops in path length (e.g., Fig. 6c) highlight the discrepancy between high-cost homotopies that are easy to sample (navigating around large obstacles) and near-optimal homotopies that are difficult to sample (crossing narrow passages through obstacles). Steady decreases in path length (e.g., Fig. 6a) show planning problems where sampling can more easily discover lower-cost homotopies.
Fig. 5: Top: Convergence Percentage across 100 trials per environment. For most environments, sampling from the IS results in a worse Convergence Percentage than GuILD for a given sample budget (leftmost curve is better). Bottom: Bonferroni-corrected nonparametric confidence intervals (CI) for the median Sample Efficiency. For all but one GuILD instance across environments, the median Sample Efficiency is smaller than that of the IS with 95% confidence. Bandit GuILD consistently ranks among the best densification strategies.

Fig. 6: Normalized Path Cost, median across 100 trials. Sampling with GuILD reaches a near-optimal path length more quickly than sampling from the IS, with significant improvement in all but the easiest planning domain (leftmost curve is better).

GuILD is supported across all environments: sampling with GuILD consistently yields a lower cost than sampling with IS. In planning problems with low-cost homotopies, GuILD automatically focuses sampling to deliver paths through these narrow passages. With the Trap environment, GuILD instances identify the narrow passage around 1100-1200 samples and quickly optimize within homotopies crossing the passage. By contrast, the IS baseline requires nearly double the samples to find the passage. The TwoWall environment shows a similar trend, although with additional intermediate-cost homotopies that only traverse one of the two narrow passages (Fig. 6b).

The SE2Maze and HERB Shelf demonstrate the anytime performance of GuILD on environments with less distinct homotopy costs. As a result, all sampling schemes steadily improve path cost, similar to Forest. However, in HERB Shelf, GuILD instances have a much steeper improvement, suggesting that GuILD may yield more significant sample efficiency improvements in higher dimensions.

VI. CONCLUSION

GuILD is a new framework for incremental densification with a simple insight: even when the planner fails to discover a shorter path, the search tree still contains valuable information that can immediately improve the densification strategy. GuILD selects a beacon from the search tree that decomposes the original sampling/planning problem into two smaller subproblems. Improving the cost-to-come for any beacon allows GuILD to adaptively shrink and expand the Local Subsets that it samples from. Similar to the Informed Set that GuILD builds upon, Local Subsets can be easily (and efficiently) incorporated into any sampling-based optimal motion planning algorithm.

Even simple beacon selectors, such as Uniform, can dramatically accelerate the convergence rate relative to the Informed Set densification baseline. We also propose a Bandit selector using EXP3, an adversarial bandit algorithm that consistently ranks among the best beacon selectors across all the planning problems we considered. In particular, GuILD excels in domains with difficult-to-sample homotopy classes and high-dimensional planning problems.

In this work, we have primarily considered either heuristic beacon selectors (Greedy) or beacon selectors that learn from experience online (Bandit). Better heuristics for selecting beacons may exist, including ones that build on prior work in identifying and sampling bottleneck points. Embedding these approaches within the GuILD framework retains theoretical guarantees of asymptotic optimality, and simplifies the learning problem from generating focused samples to selecting beacons that yield focused samples.

REFERENCES

[1] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” I. J. Robotics Res., vol. 30, no. 7, pp. 846–894, 2011.
[2] ——, “Incremental sampling-based algorithms for optimal motion planning,” in RSS, 2010.
[3] O. Arslan and P. Tsiotras, “Use of relaxation methods in sampling-based algorithms for optimal motion planning,” in *ICRA*, 2013, pp. 2421–2428.

[4] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, “Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs,” in *ICRA*, 2015, pp. 3067–3074.

[5] M. P. Strub and J. D. Gammell, “Advanced BIT* (ABIT*): Sampling-based planning with advanced graph-search techniques,” in *ICRA*, 2020, pp. 130–136.

[6] E. Schmerling, T. W. E. Lee, and A. Faust, “Adaptively Informed Trees (AIT*)**: Fast asymptotically optimal path planning through adaptive heuristics,” in *ICRA*, 2020, pp. 3191–3198.

[7] S. Koenig, M. Likhachev, and D. Furcy, “Lifelong planning A*,” in *Artif. Intell.*, vol. 155, nos. 1-2, pp. 93–146, 2004.

[8] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, “Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic,” in *IROS*, 2014, pp. 2997–3004.

[9] D. Hsu, T. Jiang, J. Reif, and Z. Sun, “The bridge test for sampling narrow passages with probabilistic roadmap planners,” in *ICRA*, 2003, pp. 4420–4426.

[10] B. Ichter, J. Harrison, and M. Pavone, “Learning sampling distributions for robot motion planning,” in *ICRA*, 2018, pp. 7087–7094.

[11] R. Kumar, A. Mandalika, S. Choudhury, and S. Srinivasa, “LEGO: Leveraging experience in roadmap generation for sampling-based planning,” in *IROS*, 2019, pp. 1488–1495.

[12] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, “Learned critical probabilistic roadmaps for robotic motion planning,” in *ICRA*, 2020, pp. 9535–9541.

[13] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[14] S. M. LaValle and J. J. Kuffner, “Randomized kinodynamic planning,” *I. J. Robotics Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[15] J. J. Kuffner and S. M. LaValle, “RRT-Connect: An efficient approach to single-query path planning,” in *ICRA*, 2000, pp. 995–1001.

[16] D. Hsu, J.-C. Latombe, and R. Motwani, “Path planning in expansive configuration spaces,” in *ICRA*, 1997, pp. 2719–2726.

[17] L. Janson, E. Schmerling, A. Clark, and M. Pavone, “Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions,” *I. J. Robotics Res.*, vol. 34, no. 7, pp. 883–921, 2015.

[18] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, “MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space,” in *ICRA*, 1999, pp. 1024–1031.

[19] C. Holleman and L. E. Kavraki, “A framework for using the workspace medial axis in PRM planners,” in *ICRA*, 2000, pp. 1408–1413.

[20] H. Kurniawati and D. Hsu, “Workspace importance sampling for probabilistic roadmap planning,” in *IROS*, 2004, pp. 1618–1623.

[21] Y. Yang and O. Brock, “Adapting the sampling distribution in PRM planners based on an approximated medial axis,” in *ICRA*, 2004, pp. 4405–4410.

[22] J. J. Ryan and M. H. Overmars, “Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners,” in *I. J. Robotics Res.*, vol. 24, no. 12, pp. 1055–1071, 2005.

[23] J. Denny, E. Greco, S. Thomas, and N. M. Amato, “MARRT: Medial axis biased rapidly-exploring random trees,” in *ICRA*, 2014, pp. 90–97.

[24] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, “OBPRM: An obstacle-based PRM for 3D workspaces,” in *WAFR*, 1998, pp. 155–168.

[25] V. Boor, M. Overmars, and A. van der Stappen, “The gaussian sampling strategy for probabilistic roadmap planners,” in *ICRA*, 1999, pp. 1018–1023.

[26] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato, “An obstacle-based rapidly-exploring random tree,” in *ICRA*, 2006, pp. 895–900.

[27] L. Kramer, W. Granzer, and W. Kastner, “A new approach for robot motion planning using rapidly-exploring randomized trees,” in *Fifth IEEE International Conference on Industrial Informatics*. IEEE, 2011, pp. 263–268.

[28] S. Kiesel, E. Burns, and W. Ruml, “Abstraction-guided sampling for motion planning,” in *International Symposium on Combinatorial Search*, vol. 3, no. 1, 2012.

[29] C. Chamzas, A. Shrivastava, and L. E. Kavraki, “Using local experiences for global motion planning,” in *ICRA*, 2019, pp. 8606–8612.