Dynamic Region-biased Rapidly-exploring Random Trees

Jory Denny[1], Read Sandström[2], Andrew Bregger[2], and Nancy M. Amato[2]

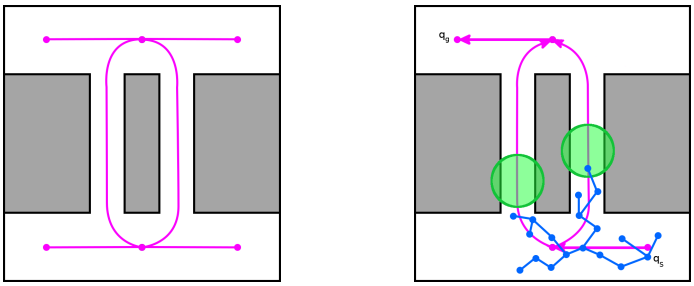
[1] Department of Mathematics and Computer Science,
University of Richmond, Richmond, VA, USA,
jdenny@richmond.edu.
[2] Parasol Lab, Department of Computer Science and Engineering,
Texas A&M University, College Station, TX, USA,
{readamus, amato}@cse.tamu.edu.

Abstract. Current state-of-the-art motion planners rely on sampling-based planning to rapidly cover and explore the problem space for a solution. However, sampling valid configurations in narrow or cluttered workspaces remains a challenge because the free space is relatively restricted. If a valid path for the robot is highly correlated with a path in the workspace, then the planning process would benefit from a representation of the workspace that captured its salient topological features. Prior approaches have investigated exploiting geometric decompositions of the workspace to bias sampling; while beneficial in some environments, complex narrow passages remain challenging to navigate.
In this work, we present Dynamic Region-biased RRT, a novel sampling-based planner that guides the exploration of a Rapidly-exploring Random Tree (RRT) by dynamically moving sampling regions along an embedded graph that captures the workspace topology. These sampling regions are dynamically created, manipulated, and destroyed to greedily bias sampling through unexplored passages that lead to the goal. We compare our approach with related methods on a set of maze-like problems.

1 Introduction

State-of-the-art motion planners rely on randomized sampling [19] to construct an approximate model of the problem space which in turn is searched for a valid path. Planners for various types of robotic systems rely on this paradigm. However, narrowness or clutter in the space often provides a fundamental challenge for these planners [13].

Motion planning applications such as robotics, virtual prototyping [35], and gaming/virtual reality [21] often require an object to move through narrow areas of an environment. When a space is sufficiently narrow that the probability of sampling a valid object position is low, it is deemed a "narrow passage." This limited free space implies that a collision-free path through the passage will likely be highly correlated with the passage geometry.



(a) Embedded Graph (b) Dynamic Region RRT

Fig. 1. Example embedding graph (a) and execution of Dynamic Region-biased RRT (b) are shown. The embedding graph encodes multiple paths of exploration through the space. These are explored by workspace regions that bias RRT growth in $\mathcal{C}_{space}$.

Based on the relationship between workspace geometry and allowable solutions, many approaches extract information from the workspace to influence the planning process. Examples include the use of collision information to adapt sampling [5, 36], directing sampling with obstacle geometry [30, 37], and decomposing the world into subproblems for the planner [18, 25].

In this paper, we study how a representation of the workspace topology can be employed to create a dynamic sampling strategy that leads a planner through the workspace and focuses its resources on unexplored areas. We present a novel planner called Dynamic Region-biased RRT based on Rapidly-exploring Random Trees (RRTs). Our method begins by pre-computing an embedding graph that captures the topology of the free workspace (Figure 1(a)). At planning time, we compute a flow along the embedding graph extending outward from the start configuration. Then, the method dynamically creates and moves sampling regions along the flow to direct RRT exploration through the environment (Figure 1(b)). We use a Reeb Graph [28] as our embedding graph because it is a compact encoding of the changes in the topology of a volumetric space. Our contributions include:

- a novel workspace guidance for RRTs, called Dynamic Region-biased RRT, whereby sampling is biased toward dynamic regions moving along an embedding graph in workspace,
- the first sampling-based planner exploiting a Reeb Graph of the free workspace,
- and an experimental validation of our approach.

To the best of our knowledge, our approach is the first fully automated RRT approach that biases growth based on dynamically moving regions and can be used as a framework to wrap around many RRT methods.

2 Preliminaries and Related Work

In this section, we define important preliminaries of motion planning and describe relevant related work to our novel planner.

2.1 Motion Planning Preliminaries

In this paper, we discuss motion planning in the context of holonomic robots, i.e., robots whose *degrees of freedom* (DOFs) are fully controllable. The DOFs for a robot parameterize its position and orientation in the 2-d or 3-d world, or *workspace*. They include, for example, object position, orientation, joint angles, etc. A configuration is a single specification of the DOFs $q = \langle x_1, x_2, \ldots, x_n \rangle$, where x_i is the ith DOF and n is the total number of DOFs. The set of all possible parameterizations is called the *configuration space ($\mathcal{C}_{space}$)* [23].

$\mathcal{C}_{space}$ is often partitioned into two subsets, *free space* ($\mathcal{C}_{free}$) and *obstacle space* ($\mathcal{C}_{obst}$). In general, it is infeasible to explicitly compute a representation of $\mathcal{C}_{obst}$ [29], but we can often determine the validity of a configuration q quite efficiently by performing a workspace collision test between the robot placed at q and the environment. If the robot placed at q does not collide with itself or the environment, q is a valid configuration and $q \in \mathcal{C}_{free}$.

Given a start configuration and a goal configuration or region, the motion planning problem is the problem of finding a continuous trajectory in $\mathcal{C}_{free}$ between the start and goal. We define a *query* as a start and goal pair.

While we describe our planner in terms of holonomic robots, i.e., planning in $\mathcal{C}_{space}$ only, it can be easily extended to nonholonomic robots, similar to [19].

Regions. We define a *region* as any bounded volume in the workspace. Common examples include axis-aligned bounding boxes (AABBs) and bounding spheres (BSs) — note that each point p of a region R maps to a possibly infinite number of configurations in $\mathcal{C}_{space}$, e.g., by placing the center of mass of the robot at p and randomizing the remaining DOFs. The usage of bounding volumes is not unique to motion planning and can be found in many fields, e.g., collision detection libraries [22].

Homotopy. Two paths are defined to be *homotopy equivalent* if and only if one path can be continuously deformed to the other without transitioning through an obstacle region. A *homotopy class* is a set of paths such that all pairs are homotopy equivalent.

2.2 Sampling-based Planning

The high complexity of motion planning [29] has driven research toward developing sampling-based techniques that efficiently explore $\mathcal{C}_{free}$ in search of valid paths. Sampling-based planners, e.g., Rapidly-exploring Random Trees (RRTs) [19] attempt to construct a graph, called a *roadmap*, that is an approximate model of $\mathcal{C}_{free}$. We focus our related work on RRT approaches as they are most relevant to our planner.

Rapidly-exploring Random Trees (RRTs) Rapidly-exploring Random Trees (RRTs) [19] belong to the class of planning approaches tailored for solving single-query motion planning problems. RRTs iteratively grow a tree outwards from a root configuration q_{root}. In each expansion attempt, a random configuration q_{rand} is chosen, and the nearest configuration within the tree q_{near} is extended towards q_{rand} up to or at a fixed step size Δq [16]. From this extension, a new configuration q_{new} is computed and added to the tree if and only if there is a valid path from q_{near} to q_{new}.

RRTs have been quite useful for a broad range of robotic systems including dynamic environments [10], kinematically constrained robots [32], and nonholonomic robots [19]. As an example, to extend the algorithm for nonholomic systems, a random (or best) robot control is selected to steer q_{near} toward q_{rand} instead of stepping in a straight-line in $\mathcal{C}_{space}$ – commonly denoted x_{near}, x_{rand}, and $\mathcal{X}_{space}$ (state space) for systems containing higher order components, e.g., velocity. RRTs are typically probabilistically complete, and their Voronoi bias allows them to converge exponentially to the uniform sampling distribution over $\mathcal{C}_{free}$. This implies that RRTs explore $\mathcal{C}_{free}$ effectively. Despite these important properties, RRTs lack efficiency in navigating many types of narrow passages. In this section, we highlight a few RRT variants relevant to this research.

A few approaches attempt to limit needless RRT extensions. RRT with collision tendency [5] tracks the controls that have been tried when extending a node to reduce duplicated computations. Once all inputs have been tried, the node is excluded from nearest neighbor selection. Dynamic-Domain RRT [36] biases the random node selection to be within a radius r of q_{near} or expansion will not occur. The radius is dynamically determined from failed expansion attempts. RRT-Blossom [14] uses the idea of regression constraints to only add edges to a tree which explore new portions of the space and proposes a flood-fill approach when expanding from a node.

Obstacle-based RRT (OBRRT) [30] and Retraction-based RRT [37] exploit obstacle information to direct RRT growth along obstacle surfaces to improve RRT performance in narrow passages.

Recent approaches, e.g., RRT* [15], attempt to find optimal or near optimal motions. RRT* expands in the same way as RRT except after expansion the tree will locally "rewire" itself to ensure optimization of a cost function. RRT* has been shown to be quite effective in asymptotically finding shortest paths. In practice, RRT* requires many iterations to produce near optimal solutions. Recently proposed, Stable Sparse-RRT [20], a near-optimal planner, relaxes the optimality guarantee with a sparse tree to overcome this practical challenge.

Workspace-biased Planners Many planners use workspace information to aid in the planning process. Here we describe a few.

Feature Sensitive Motion Planning [25] recursively subdivides the space into "homogeneous" regions (regions of the environment containing similar properties, e.g., free or clutter), individually constructs a roadmap in each region, and merges them together to solve the aggregate problem. This framework adaptively

decides the specific planning mechanism to map to each homogeneous region, e.g., use obstacle-based sampling [1] in cluttered regions and uniform sampling in open regions.

Other approaches utilize workspace decompositions to find narrow or difficult areas of the workspace to bias $\mathcal{C}_{space}$ sampling [2, 17, 18, 27]. These methods begin by decomposing the workspace using an adaptive cell decomposition [2] or a tetrahedralization [17, 18], and then they weight the decomposition to bias sampling. However, by automatically identifying regions and disallowing dynamic region specification and modification, the planner often oversamples in fully covered regions.

A recent planning approach allows a user to define and manipulate regions of the workspace to bias probabilistic roadmap construction [7]. We utilize a similar concept of workspace regions, but do not rely on a human operator to direct region manipulation. However, we note that the user-guided approach might be more suitable in some applications.

3 Dynamic Region RRT

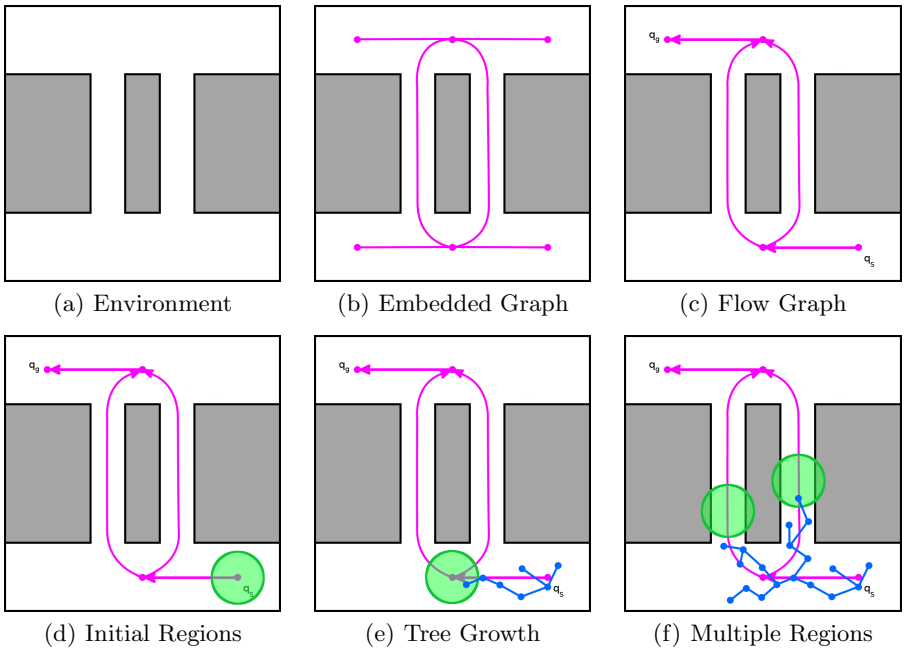
In this section, we explore a fully automated planning algorithm that was inspired by the human collaboration techniques in [7].

3.1 Algorithm Overview

Our algorithm, Dynamic Region-biased RRT, is depicted in Algorithm 1 and Figure 2. The core idea is to compute a representation of the workspace topology that describes a path through each homotopy class, and then use dynamic sampling regions to automatically guide an RRT planner along those paths.

The workspace representation is generated in the form of a spatially embedded undirected graph, called the *Embedding Graph* (Definition 1 and magenta in Figure 2(b)), as a pre-computation step in line 1. Once the query is known, we compute a directed version of the embedding graph in line 2, called the *Flow Graph* (Definition 2 and magenta in Figure 2(c)), that encodes the exploration directions leading through the workspace and toward the query goal.

Our planner then creates a sampling region (green) at the Flow Graph node that is nearest to the query start in line 4 and begins RRT growth (green in Figure 2(d)). On each iteration (line 5), the planner chooses between all of the available sampling regions and the entire environment with uniform random probability, and generates the next growth target q_{rand} from within that volume of workspace. When a new configuration is placed inside a dynamic sampling region, that region is advanced along the Flow Graph edge until it no longer contains the new sample. The net effect is that the RRT trees are biased to follow these sampling regions along the homotopy distinct workspace paths that are captured in the Flow Graph. During this process our planner modifies, creates, and destroys sampling regions in the workspace that bias RRT growth in $\mathcal{C}_{space}$. As an example, Figure 2(e) shows how the regions bias tree growth (blue) and



(a) Environment (b) Embedded Graph (c) Flow Graph

(d) Initial Regions (e) Tree Growth (f) Multiple Regions

Fig. 2. Example execution of Dynamic Region-biased RRT: (a) environment; (b) pre-computed embedding graph (magenta) of the workspace; (c) flow graph (magenta) computed from start configuration; (d) initial region (green) placed at the source vertex of the flow graph; (e) region-biased RRT growth (blue); and (f) multiple active regions (green) guiding the tree (blue) among multiple embedded arcs of the flow graph (magenta) is shown.

reach the next node of the flow graph, creating two more regions to traverse the outgoing edges from that node as shown in Figure 2(f). The algorithm continues in this manner until a stopping criteria is met (line 5), e.g., query is solved or maximum number of iterations is met.

We note that this framework is amenable to many RRT-based planners because the only interface requirement is the ability to designate the sampling boundary for each iteration.

3.2 Embedding Graph

Dynamic Region-biased RRT begins by pre-computing an *Embedding Graph* of the free workspace:

Definition 1. *An* Embedding Graph *is a 1-skeleton of the free workspace represented by an undirected graph that has been spatially embedded in the environment. The embedding graph must also satisfy the properties of a retraction [11], i.e., every point in the free workspace can be mapped onto the structure.*

Algorithm 1 Dynamic Region-biased RRT

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

1: $G \leftarrow$ ComputeEmbeddingGraph(e) {Pre-computation}
2: $F \leftarrow$ ComputeFlowGraph(G, q_s)
3: $T \leftarrow (\emptyset, \emptyset)$
4: $R \leftarrow$ InitialRegions(F, q_s)
5: **while** $\neg done$ **do**
6: RegionBiasedRRTGrowth(T, F, R)
7: **return** T

The embedding graph has certain requirements to maintain desirable properties. Namely, the embedding graph must be a deformation retract [11] of the free workspace. This implies that every point in the free workspace can be mapped onto the skeleton, i.e., the embedding graph represents the entire workspace and preserves its topology. This property allows the number of possible homotopy classes (distinct paths) in a motion planning problem to be estimated.

There are a few examples of possible data structures which satisfy this property. One example, a Generalized Voronoi Graph (GVG) [6] is the set of points equidistant to m obstacles in a space of dimension m. The GVG, however, is not always guaranteed to be connected. A few methods have proposed ideas to overcome this [6, 9]. Another example is a Reeb Graph [28] which represents transitions in level sets of a real-valued function on a manifold, i.e., nodes of the graph are critical values of the function, referred to as a Morse function, and edges are the topological transitions between them. It has applications in various parts of computational geometry and computer graphics such as shape matching [12], iso-surface remeshing [34], and simplification [33]. We choose a Reeb Graph as our embedding graph because it satisfies our requirements and is sufficiently simple to compute. We do not claim that this choice is optimal in any sense.

Our embedding graph algorithm, Algorithm 2, begins by computing a Delaunay tetrahedralization of the free workspace [31] in line 1. We then build a Reeb Graph from the resulting set of tetrahedra with the algorithm found in [26] at line 2. The next step embeds the Reeb graph nodes in the free workspace by shifting each node to the center of its closest tetrahedron (lines 3–5). Finally, we embed each Reeb Graph arc by searching for a path through the related tetrahedra that connects the embedded start and end node (lines 6–8).

The related tetrahedra for a Reeb Graph arc are the set of connected decomposition tetrahedra that are found between the topological critical points (which are the Reeb Graph nodes). This is determined during the Reeb Graph construction: critical points occur whenever there are two or more adjacent tetrahedra that have a pair of vertices that cannot be connected by a straight-line through free space. This represents a divide in the freespace volume: thus, each Reeb Graph arc captures a distinct workspace homotopy class between two embedded Reeb nodes, and the embedded arc represents a specific (workspace) path in that

Algorithm 2 Compute Embedding Graph

Input: Environment e
Output: Embedding Graph G
1: $D \leftarrow$ TETRAHEDRALIZATION(e)
2: $R = (R_V, R_E) \leftarrow$ CONSTRUCTREEBGRAPH(D)
3: $G = (V, E) \leftarrow (\emptyset, \emptyset)$
4: **for all** $v \in R_V$ **do**
5: $\quad V \leftarrow V \cup \{D.$CLOSESTTETRAHEDRON$(v)\}$
6: **for all** $e \in R_E$ **do**
7: $\quad s \leftarrow e.$SOURCE$(); t \leftarrow e.$TARGET$()$
8: $\quad E \leftarrow E \cup (s, t, D.$FINDPATH$(s, t))$
9: **return** G

class. We ensure that the embedded arcs lie entirely within the free workspace by connecting adjacent tetrahedra through the center of their adjacent faces (as opposed to connecting the tetrahedra centers directly). In this way, the final embedding graph is a set of points and polygonal chains spatially contained in the free workspace, i.e., it is a retraction of the free workspace.

Additionally, we note that every workspace homotopy is a generalization of one or more $\mathcal{C}_{space}$ homotopies. This is easy to see by considering that the projection of a $\mathcal{C}_{space}$ homotopy into workspace is simply the positional DOFs. Thus, by representing all workspace homotopies, the embedding graph contains a partial representation of each $\mathcal{C}_{space}$ homotopy.

3.3 Flow Graph

Once the Embedding Graph is computed and a query is posed, our planning algorithm computes a *Flow Graph* from the embedding graph:

Definition 2. *A* Flow Graph, $F = (V, E)$ *is a directed subset of the embedding graph in which the edges point along the shortest path toward the node* $v \in V$ *that is nearest to the query goal.*

To compute the flow graph, we find the closest embedding graph vertex to the start configuration and perform a breadth-first search (BFS) traversal of the embedding graph. We include cross-edges of the BFS traversal in the directed graph directed opposite of the earlier discovered node. Then, we traverse the graph a second time: starting from the embedding graph vertex that is closest to the goal, we backtrack along the inbound edges to discover all of that vertex's ancestors. We then prune from the Flow Graph any node that was not discovered in the this second pass, thereby removing edges that do not lead toward the goal area. The resulting Flow graph is used to coordinate region construction, modification, and deletion.

Algorithm 3 Region-biased RRTGrowth

Input: Tree T, Flow Graph F, Regions R
Require: ϵ is a multiple of the robot radius, τ is a maximum for failed extension
{Region-biased RRT extension}
1: $r \leftarrow$ SELECTREGION(R)
2: $q_{rand} \leftarrow r$.GETRANDOMCFG()
3: $q_{near} \leftarrow$ NEARESTNEIGHBOR(T, q_{rand})
4: $q_{new} \leftarrow$ EXTEND($q_{near}, q_{rand}, \Delta$)
{Advance regions along Flow edges}
5: **for** $r \in R$ **do**
6: **while** r.INREGION(q_{new}) **do**
7: r.ADVANCEALONGFLOWEDGE()
8: **if** r.ATENDOFFLOWEDGE() **then**
9: $R \leftarrow R \setminus \{r\}$
{Delete useless regions}
10: **if** r.NUMFAILURES() $> \tau$ **then**
11: $R \leftarrow R \setminus \{r\}$
{Create new regions}
12: **for all** $v \in F$.UNEXPLOREDVERTICES() **do**
13: **if** $\delta(v, q_{new}) < \epsilon$ **then**
14: $R \leftarrow R \cup$ NEWREGION(v)
15: F.MARKEXPLORED(v)

3.4 Region-biased RRT Growth

Algorithm 3 shows our RRT growth strategy. The algorithm takes a tree T, a flow graph F, and a set of regions R as input and combines four phases. In the first phase, the algorithm performs a region-biased RRT extension (lines 1–4). It selects a random region and picks a configuration q in the region by randomly selecting a point in the region to initialize the positional DOFs and randomizes the remaining DOFs of q. Note, an invisible region encompasses the entire environment to maintain probabilistic completeness as in the collaborative algorithm presented in [7]. In the second phase, we advance the regions along their associated flow graph edges if needed (lines 5–9). To do this, we determine if the extended node q_{new} reached some portion of any region r. If so, we move each such region r to the next point on its embedded flow edge in the workspace until it no longer contains q_{new}. When a region reaches the end of its flow edge, it is deleted. The third phase deletes fruitless regions determined by some threshold τ of consecutive failed tree extensions (lines 10–11). In the final phase, we create new regions if q_{new} is within an ϵ distance of an unexplored flow vertex v (lines 12–15). We create a region for each outgoing edge of v and mark v as explored on the flow graph.

Our algorithm has little runtime overhead compared with standard RRT growth. It comes in the form of selecting, advancing, creating, and deleting regions. The number of regions is bounded by $O(|F_E|)$, where F_E is the set of edges in the flow graph F. Our expansion step has an overhead of $O(|F_E|+|F_V|)$

where F_V is the set of vertices of the flow graph F. The $O(|F_E|)$ term stems from the requirement to check if a new extension has reached any active region, while the $O(|F_V|)$ term comes from checking whether a new extension has reached an unvisited flow graph node. Although this is an extreme upper bound, in practice the number of active regions is much smaller than the number of flow graph edges, and the number of unvisited vertices drops monotonically during execution. In the future, we can look to use a KD-tree to try and improve this bound to $O\left(|F_E| + log(|F_V|)\right)$.

3.5 Discussion

Our algorithm is able to capture the salient topological features of the free workspace. In problems where $\mathcal{C}_{space}$ is strongly related to the workspace, our algorithm provides a convenient methodology to explore the space.

Some problems exhibit only a partial correlation between $\mathcal{C}_{space}$ and workspace. One example is manipulation planning, where end-effector's path is strongly correlated with the environment, but the rest of the robot may not be. In such cases, an embedding graph could be used to guide sampling for the correlated component. This would additionally require a sampling method that is able to place the correlated component first and subsequently sample the remaining DOFs, such as reachable volumes [24] or cyclic coordinate descent [4].

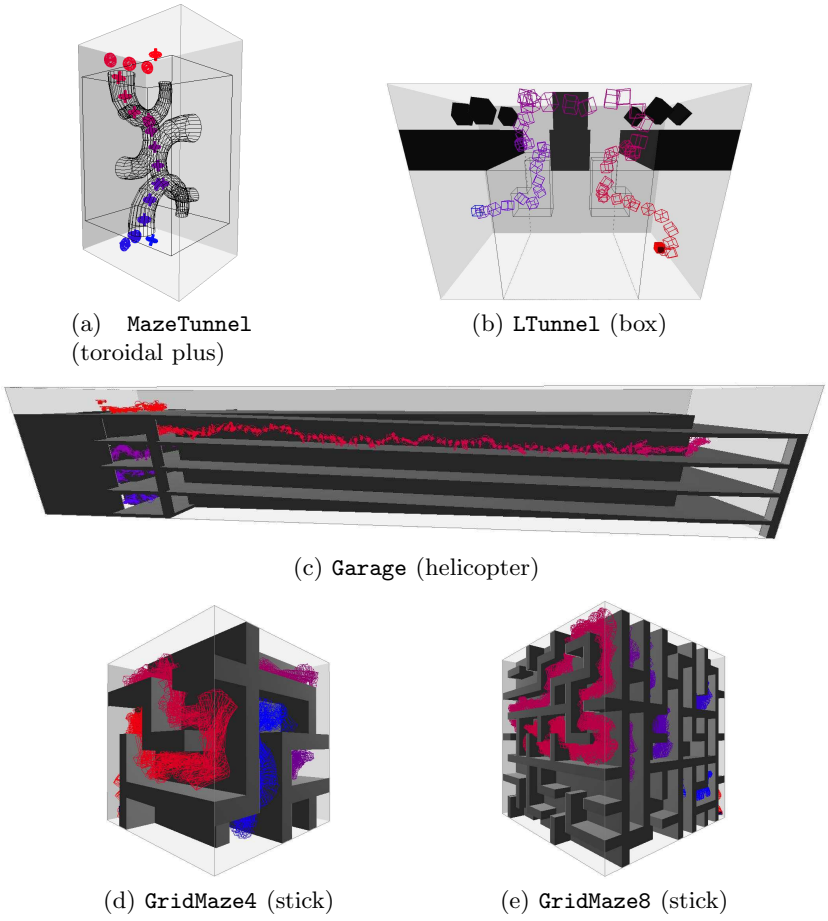
Unfortunately, not all problems exhibit such a correlation. In problems such as protein folding, the use of an embedding graph may not be sensible. However the underlying idea may still apply, which is to use regions and topological analysis in low-dimensional manifolds of $\mathcal{C}_{space}$ to explore $\mathcal{C}_{free}$. Dynamic Region-biased RRT uses the workspace itself as such a manifold, but one could feasibly design alternatives for other classes of problems.

4 Experimental Analysis

All methods were implemented in a C++ motion planning library developed in the Parasol Lab at Texas A&M University. It uses a distributed graph data structure from the Standard Template Adaptive Parallel Library (STAPL) [3], a C++ library designed for parallel computing.

All experiments were executed on a latop running Fedora23 with an Intel(R) Core(TM) i7-6500U CPU at 2.5 GHz, 8 GB of RAM, and the GNU gcc compiler version 5.3.1.

We compared Dynamic Region-biased RRT (DRRRT) against RRT [19], Dynamic-Domain RRT [36], SyCLoP with RRT [27], and the Probabilistic RoadMap (PRM) approach using Workspace Importance Sampling (WIS) [17]. Like DRRRT, Dynamic-Domain RRT and SyCLoP both use adaptive sampling distributions. SyCLoP and WIS also employ pre-computations to decompose the workspace. SyCLoP additionally uses a search over the decomposition graph to guide an RRT.



(a) `MazeTunnel` (toroidal plus)

(b) `LTunnel` (box)

(c) `Garage` (helicopter)

(d) `GridMaze4` (stick)

(e) `GridMaze8` (stick)

Fig. 3. The experiment environments and (robot). The `Garage` contains multiple alternative paths, and the `GridMaze`s are 3D mazes with internal pathways and tunnels throughout. The robots in all cases are 6 DOF rigid bodies. In the `GridMaze`s, the robot length is equal to the tunnel width to make turning more difficult.

We demonstrate our algorithm on holonomic robots in five environments, including `MazeTunnel`, `LTunnel`, `Garage`, `GridMaze4`, and `GridMaze8` as shown in Figure 3. We also evaluate performance on the `LTunnel` with nonholonomic robots, comparing against RRT and SyCLoP, as a proof-of-concept that the embedding graph can benefit nonholonomic systems. These environments contain narrow passages in the workspace that are correlated with valid paths in $\mathcal{C}_{space}$, and are thus good exhibitions of our planner's ability to exploit that correlation. `MazeTunnel` exhibits false passages while `LTunnel` exhibits narrow entrances. `Garage` has multiple homotopy classes and a thin-walled maze. The `GridMaze` environments have long, winding paths in a cramped tunnel that hinders the

robot's ability to change orientation. The robots in all cases are 6 DOF rigid bodies.

We use a Δq value that is approximately 5-10% of the diagonal of each environment. The nonholonomic experiment uses a fully actuated robot with a 1:2 ratio of random vs. best control selection and a 1:2 ratio of fixed vs. variable timestep for extension [19].

Each experiment ran until the query was solved, or 20k nodes had been added to the roadmap, or 20 minutes had elapsed. We performed 35 trials for each experiment and removed the fastest and slowest run from each. Average run times and standard deviations are shown in Figure 4.

4.1 Discussion

As Figure 4 shows, our planner exhibits faster online planning time as compared with the other methods in the holonomic problems. The improvements over the closest competitors are significant, with p-values of .0083 against RRT and .0175 against Dynamic-Domain RRT in `MazeTunnel` (using the student's unpaired t-test). Neither RRT nor Dynamic-Domain RRT was able to solve the `Garage` problem. WIS showed strong performance in `LTunnel` and `Garage`, but lacked efficiency in `MazeTunnel` and consisitency in `GridMaze4`. Only DRRRT was able to solve the large `GridMaze8`, which pushes the algorithm to the limits of what it can handle with high reliability.

Table 1. Success rates in each experiment.

Planner	DRRRT	RRT	Dynamic-Domain RRT	SyCLoP-RRT	WIS
`MazeTunnel`	100%	100%	100%	100%	100%
`LTunnel`	100%	100%	100%	100%	100%
`Garage`	100%	0%	0%	100%	100%
`GridMaze4`	100%	100%	100%	100%	100%
`GridMaze8`	76%	0%	0%	0%	0%
`LTunnel` (nonholonomic)	90%	24%	-	97%	-

In the nonholonomic experiment, DRRRT was faster than RRT (p-value less than .0001) and slower than SyCLoP (p-value .0002) in `LTunnel` with high confidence. While DRRRT improves over RRT, it doesn't perform as well as SyCLoP for this problem. This is because SyCLoP doesn't always expand from the front of the tree, which helps prevent it from getting stuck when the leading nodes in the tree are difficult to extend from. Emulating this aspect of SyCLoP leaves a promising future refinement for DRRRT.

In terms of pre-computation time, our embedding graph took on average 3.5 seconds to build in `MazeTunnel`, 0.7 seconds in `LTunnel`, and 2.3 seconds in `Garage`. It is important to note that the Reeb Graph computational efficiency is $O(n^2)$ in the worst case where n is the number of points in the tetrahedralization [26]. In other words, the Reeb Graph computation is significantly more

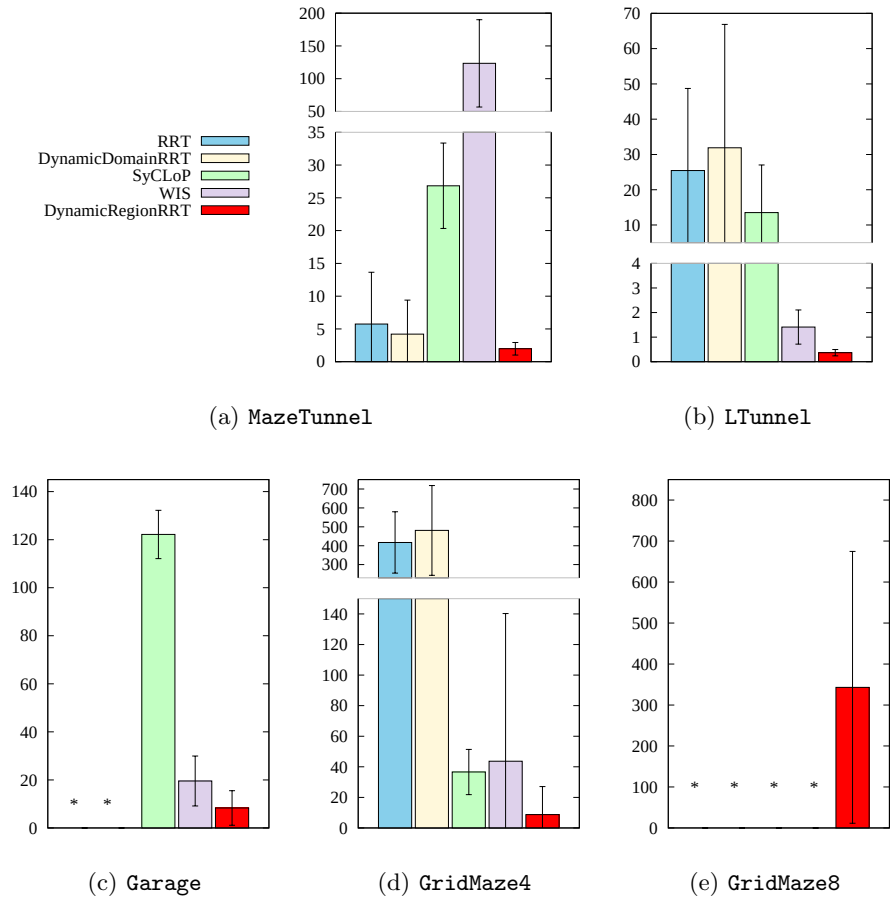


Fig. 4. On-line planning time comparing Dynamic Region-biased RRT with RRT, Dynamic-Domain RRT, WIS, and SyCLoP in holonomic problems. The values are the average times over 33 trials. Error bars indicate standard deviation. A * indicates that none of the trials solved the query.

efficient in simpler environments. It is possible to reduce this computation time to a little worse than $O(n \log n)$ with a more sophisticated construction algorithm [8], which we leave to future work.

Beyond this, we would like to make a special note of the dynamic aspect of our regions. In these test environments, there are false passages in the workspace. Our algorithm may explore them while sampling from the whole environment, but its region-guided efforts are concentrated on potentially viable paths in the workspace and avoid the false passages. However, if a large number of consecutive attempts to sample within a region fail, we delete the region as it is likely a path that is only viable for workspace and not for the robot's $\mathcal{C}_{space}$. In this way,

our algorithm is robust to very complex environments with multiple homotopy classes in the workspace.

5 Conclusion

In this paper, we introduced a new adaptive, workspace-based sampling strategy for RRTs and demonstrated its use in holonomic and nonholonomic problems. The key development is the use of dynamic regions with an embedding graph to represent the workspace structure as a biasing strategy for sampling-based planners. While there are environmentally dependent trade-offs involved, this method demonstrates benefits in problems where valid paths are strongly correlated with the workspace geometry. Additionally, the embedding graph offers an intuitive means to bias nonholonomic problems with workspace information.

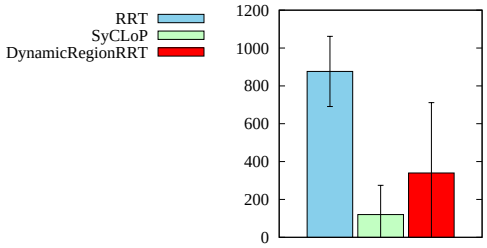


Fig. 5. On-line planning time comparing Dynamic Region-biased RRT with RRT and SyCLoP in the nonholonomic `LTunnel`. Error bars indicate standard deviation.

Future directions for this work include studying the effects of using an embedding graph with PRMs and optimal planners, applying more advanced Reeb Graph algorithms or approximate embedding graph constructions, and improving performance on nonholonomic problems.

Acknowledgements

This research supported in part by NSF awards CNS-0551685, CCF-0833199, CCF-1423111, CCF-0830753, IIS-0916053, IIS-0917266, EFRI-1240483, RI-1217991, by NIH NCI R25 CA090301-11. J. Denny was supported in part by an NSF Graduate Research Fellowship during the main portion of this research.

References

1. Amato, N.M., Bayazit, O.B., Dale, L.K., Jones, C., Vallejo, D.: OBPRM: an obstacle-based PRM for 3d workspaces. In: Proceedings of the third Workshop on the Algorithmic Foundations of Robotics. pp. 155–168. A. K. Peters, Ltd., Natick, MA, USA (1998), (WAFR '98)
2. van den Berg, J.P., Overmars, M.H.: Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. Int. J. Robot. Res. 24(12), 1055–1071 (2005)

3. Buss, A.A., Harshvardhan, Papadopoulos, I., Pearce, O., Smith, T.G., Tanase, G., Thomas, N., Xu, X., Bianco, M., Amato, N.M., Rauchwerger, L.: STAPL: standard template adaptive parallel library. In: Proceedings of of SYSTOR 2010: The 3rd Annual Haifa Experimental Systems Conference, Haifa, Israel, May 24-26, 2010. pp. 1–10. ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1815695.1815713
4. Canutescu, A.A., Dunbrack, Jr., R.L.: Cyclic coordinate descent: A robotics algorithm for protein loop closure. Protein Sci. 12(5), 963–972 (2003)
5. Cheng, P., LaValle, S.: Reducing metric sensitivity in randomized trajectory design. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS). vol. 1, pp. 43–48 vol.1 (2001)
6. Choset, H., Burdick, J.: Sensor-based exploration: The hierarchial generalized voronoi graph. Int. J. Robot. Res. 19(2), 96–125 (2000)
7. Denny, J., Sandstrom, R., Julian, N., Amato, N.M.: A region-based strategy for collaborative roadmap construction. In: Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR). Istanbul, Turkey (August 2014)
8. Doraiswamy, H., Natarajan, V.: Efficient algorithms for computing reeb graphs. Comput. Geom. Theory Appl. 42(6-7), 606–616 (Aug 2009), http://dx.doi.org/10.1016/j.comgeo.2008.12.003
9. Foskey, M., Garber, M., Lin, M.C., Manocha, D.: A voronoi-based hybrid motion planner for rigid bodies. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS). pp. 55–60 (2001)
10. Gayle, R., Sud, A., Lin, M.C., Manocha, D.: Reactive deformation roadmaps: Motion planning of multiple robots in dynamic environments. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS) (2007)
11. Hatcher, A.: Algebraic Topology. Cambridge University Press (2001), http://www.math.cornell.edu/~hatcher/
12. Hilaga, M., Shinagawa, Y., Kohmura, T., Kunii, T.L.: Topology matching for fully automatic similarity estimation of 3d shapes. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. pp. 203–212. SIGGRAPH '01, ACM, New York, NY, USA (2001), http://doi.acm.org/10.1145/383259.383282
13. Hsu, D., Latombe, J.C., Kurniawati, H.: On the probabilistic foundations of probabilistic roadmap planning. Int. J. Robot. Res. 25, 627–643 (July 2006)
14. Kalisiak, M., van de Panne, M.: Rrt-blossom: Rrt with a local flood-fill behavior. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006. pp. 1237–1242 (May 2006)
15. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. International Journal of Robotics Research (IJRR) 30, 846–894 (2011)
16. Kuffner, J.J., LaValle, S.M.: RRT-connect: An efficient approach to single-query path planning. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). pp. 995–1001 (2000)
17. Kurniawati, H., Hsu, D.: Workspace importance sampling for probabilistic roadmap planning. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS). vol. 2, pp. 1618–1623 (sept-2 oct 2004)
18. Kurniawati, H., Hsu, D.: Workspace-based connectivity oracle - an adaptive sampling strategy for prm planning. In: Algorithmic Foundation of Robotics VII, pp. 35–51. Springer, Berlin/Heidelberg (2008), book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), New York City, 2006
19. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. 20(5), 378–400 (May 2001)

20. Li, Y., Littlefield, Z., Bekris, K.E.: Sparse Methods for Efficient Asymptotically Optimal Kinodynamic Planning, pp. 263–282. Springer International Publishing, Cham (2015), http://dx.doi.org/10.1007/978-3-319-16595-0_16
21. Lien, J.M., Pratt, E.: Interactive planning for shepherd motion (March 2009), the AAAI Spring Symposium
22. Lin, M.C.: Efficient Collision Detection for Animation and Robotics. Ph.D. thesis, University of California, Berkeley, CA (Dec 1993)
23. Lozano-Pérez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Communications of the ACM 22(10), 560–570 (October 1979)
24. McMahon, T., Thomas, S., Amato, N.M.: Reachable volume RRT. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). pp. 2977–2984. Seattle, Wa. (May 2015)
25. Morales, M., Tapia, L., Pearce, R., Rodriguez, S., Amato, N.M.: A machine learning approach for feature-sensitive motion planning. In: Algorithmic Foundations of Robotics VI, pp. 361–376. Springer Tracts in Advanced Robotics, Springer, Berlin/Heidelberg (2005), (WAFR '04)
26. Pascucci, V., Scorzelli, G., Bremer, P.T., Mascarenhas, A.: Robust on-line computation of reeb graphs: Simplicity and speed. ACM Trans. Graph. 26(3) (Jul 2007), http://doi.acm.org/10.1145/1276377.1276449
27. Plaku, E., Kavraki, L., Vardi, M.: Motion planning with dynamics by a synergistic combination of layers of planning 26(3), 469–482 (June 2010)
28. Reeb, G.: Sur les points singuliers d'une forme de pfaff complement integrable ou d'une fonction numerique. Comptes Rendus Acad. Sciences Paris 222, 847–849 (1946)
29. Reif, J.H.: Complexity of the mover's problem and generalizations. In: Proc. IEEE Symp. Foundations of Computer Science (FOCS). pp. 421–427. San Juan, Puerto Rico (October 1979)
30. Rodriguez, S., Tang, X., Lien, J.M., Amato, N.M.: An obstacle-based rapidly-exploring random tree. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (2006)
31. Si, H.: Tetgen, a delaunay-based quality tetrahedral mesh generator. ACM Trans. Math. Softw. 41(2), 11:1–11:36 (Feb 2015), http://doi.acm.org/10.1145/2629697
32. Suh, C., Kim, B., Park, F.C.: The tangent bundle RRT algorithms for constrained motion planning. In: 13th World Congress in Mechanism and Machine Science (2011)
33. Wood, Z., Hoppe, H., Desbrun, M., Schröder, P.: Removing excess topology from isosurfaces. ACM Trans. Graph. 23(2), 190–208 (Apr 2004), http://doi.acm.org/10.1145/990002.990007
34. Wood, Z.J., Schröder, P., Breen, D., Desbrun, M.: Semi-regular mesh extraction from volumes. In: Proceedings of the Conference on Visualization '00. pp. 275–282. VIS '00, IEEE Computer Society Press, Los Alamitos, CA, USA (2000), http://dl.acm.org/citation.cfm?id=375213.375254
35. Yan, Y., Poirson, E., Bennis, F.: Integrating user to minimize assembly path planning time in PLM. In: Product Lifecycle Management for Society, IFIP Advances in Information and Communication Technology, vol. 409, pp. 471–480. Springer Berlin Heidelberg (2013)
36. Yershova, A., Jaillet, L., Simeon, T., Lavalle, S.M.: Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). pp. 3856–3861 (April 2005)
37. Zhang, L., Manocha, D.: An efficient retraction-based RRT planner. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA) (2008)